

UNIVERSITE DE GENEVE

FACULTE DES SCIENCES

CENTRE INTERFACULTAIRE
DE CALCUL ELECTRONIQUE

PROFESSEUR B. LEVRAT

UN SYSTEME INTERACTIF POUR LA CREATION
ET L'INTERROGATION DE FICHIERS

THESE

présentée à la Faculté des Sciences de l'Université de Genève
pour obtenir le grade de docteur ès sciences mention informatique

par

Jean-Philippe BAUD

France

THESE N° 1620

Genève

Imprimerie de la Section Physique

1974

UN SYSTEME INTERACTIF POUR LA CREATION
ET L'INTERROGATION DE FICHIERS

THESE

présentée à la Faculté des Sciences de l'Université de Genève
pour obtenir le grade de docteur ès sciences mention informatique

par

Jean-Philippe BAUD

France

THESE N° 1620

Genève

Imprimerie de la Section Physique

1974

La faculté des sciences, sur le préavis du professeur
B. LEVRAT, Centre interfacultaire de calcul électronique,
directeur de thèse, du professeur J. HARMS et du docteur
C-A. HERITIER, autorise l'impression de la présente thèse
sans exprimer d'opinion sur les propositions qui y sont
énoncées.

Genève, le 4 juillet 1973

Le doyen
Jean Muller

Thèse N° 1620



Remerciements

Je tiens tout d'abord à exprimer ma gratitude au professeur B. Levrat pour l'intérêt qu'il a porté à mon travail, pour les encouragements qu'il m'a prodigués, pour la grande liberté qu'il m'a laissée dans le choix des options, ainsi que pour ses conseils judicieux.

Je remercie le professeur J. Harms qui a accepté de faire partie du jury de thèse et dont les suggestions m'ont été particulièrement utiles tout au long de ce travail de recherche.

Je remercie le docteur Ch.-A. Héritier qui nous a initiés aux problèmes de "temps partagé" et de performances de systèmes interactifs.

Ma reconnaissance va aussi à mes amis et collaborateurs J-M. Chenais et A. Jacquesson qui ont largement contribué à la définition et à la réalisation de ce projet.

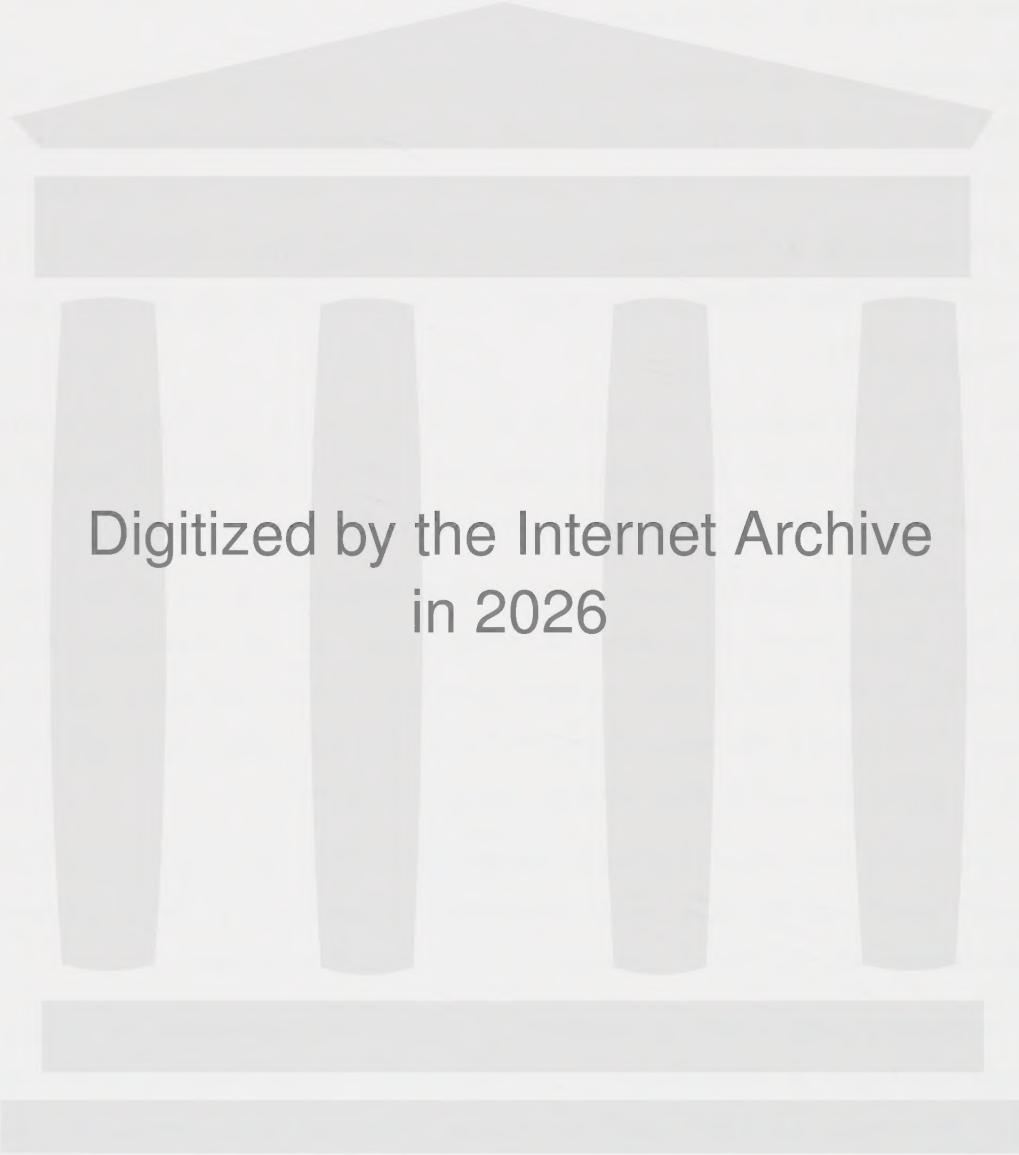
Mes remerciements vont aux responsables de la division DD du CERN, Messieurs MacLeod et Moorhead, et en particulier à Monsieur S. Santiago, notre premier utilisateur, pour toutes ses critiques et ses suggestions et la patience dont il a fait preuve lors des tests du programme.

J'exprime ma reconnaissance à tous mes collègues du CICE pour leur soutien moral, leurs conseils amicaux, leur aide désintéressée et surtout leurs critiques franches.

J'adresse mes remerciements à tous les membres du CCI, en particulier à Monsieur J-F. Renevey et à Monsieur P. Meylan qui ont mis la CDC 3800 à notre disposition pendant les heures de développement de système et les dimanches, à tous les opérateurs qui ont passé nos travaux pendant les heures d'exploitation normale.

Le projet a été financé par le Fonds National Suisse pour la Recherche Scientifique, qu'il en soit ici remercié.

Enfin, et tout spécialement, je remercie Madame L. Noël et Mademoiselle M. Vermeire pour le soin qu'elles ont apporté à la mise en page de ce texte.



Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41545902_0024

Table des matières

Introduction	1
Chapitre I : Facteurs humains	3
1. Définitions	3
2. Système interactif et programme interactif	4
3. Terminaux interactifs	5
3.1. Machines à écrire	5
3.2. Ecrans de visualisation	6
3.3. Claviers	9
3.4. Configuration utilisée	10
4. Systèmes d'exploitation	12
Chapitre II : Structure des données	15
1. Définitions	15
2. Organisation séquentielle	18
3. Organisation "random"	20
3.1. Adresse directe	20
3.2. Recherche en table	20
3.3. Calcul	21
4. Structure de liste	22
4.1. Liste simple (chaîne)	22
4.2. Fichier inversé	23
4.3. Anneaux	23
5. Structure arborescente	24
6. Organisation séquentielle indexée	25
Chapitre III : Description de l'interpréteur INFOL	27
1. Définitions	27
2. Définition et organisation des données	28
2.1. Définition du champ	28
2.2. Définition de l'enregistrement	29
2.3. Structure du fichier	32

3.	Fonctions principales du système	34
3.1.	Création du fichier	34
3.2.	Interrogation	36
3.3.	Mise à jour	39
3.4.	Autres fonctions	40
4.	Format libre	41
Chapitre IV : Programme interactif de gestion de fichiers		43
1.	Description générale de l'interpréteur	45
2.	Routine d'initialisation DEBU	46
3.	Routine de lecture GET	50
4.	Routine PHASE1	56
5.	Traitement des erreurs pendant l'INITIAL INPUT	59
6.	Option \$SOS	62
7.	Routine de sélection par clavier de fonctions	66
8.	Routine WALK	70
9.	Analyse statistique	74
Chapitre V : Performances		79
1.	Définitions	79
2.	Règles générales d'optimisation	79
3.	Segmentation du programme	80
4.	Optimisation des opérations d'entrée- sortie	82
5.	Traitement des interrogations complexes	84
6.	Les programmes interactifs et le FORTRAN	88
Chapitre VI : Conclusions et perspectives		89
Annexe I	: Listing de la sous-routine GET	91
Annexe II	: Description de la routine SPY	93
Annexe III	: Bibliographie	95

Introduction

Le concept de système généralisé de stockage et de recherche d'information n'est pas nouveau. Les premiers langages non procéduraux de traitement de fichiers sont apparus vers 1965. En particulier à cette époque, T.W. Olle a développé chez Control Data le système INFOL, acronyme anglais de "Information Oriented Language". Il a connu dès 1968 un grand succès auprès des utilisateurs du Centre de calcul de l'Université de Genève. Malheureusement, écrit en langage assembleur, il n'est installé que sur la série des ordinateurs CDC 3600-3800. Certains utilisateurs se sont rapidement inquiétés de savoir ce que deviendraient leurs applications lors d'un éventuel changement d'ordinateur.

Le Centre interfacultaire de calcul électronique a donc entrepris dès 1970 l'écriture de deux interpréteurs INFOL-2, en conservant d'une part le langage externe d'INFOL-CDC, d'autre part le traitement séquentiel des fichiers, le premier système d'exploitation TAPE SCOPE des 3800 étant orienté vers l'utilisation des bandes magnétiques.

Notre objectif principal étant la portabilité, nous devons utiliser un langage supérieur, largement répandu, comme le FORTRAN ou le COBOL. Nous avons choisi le FORTRAN : J-F. Vallee (Université de Northwestern, Chicago) [1] avait réalisé une première version destinée à une CDC 6400. La structure de données développée à Northwestern a été adoptée avec de légères modifications dans les deux versions de notre interpréteur : l'une est destinée à travailler dans un système de "traitement par lot" [2], l'autre est prévue pour fonctionner par l'intermédiaire de divers terminaux (télétype, console, écran de visualisation) [3]. Les recherches ont été poursuivies dans trois directions :

- portabilité : seules trois routines de traitement de caractères doivent être codées pour chaque machine.
- mode conversationnel.
- amélioration des performances.

Chapitre I : Facteurs humains

Durant les vingt dernières années, les constructeurs d'ordinateurs ont développé des systèmes permettant de traiter les applications les plus diverses, si on exclut quelques machines spécialisées. Par contre, les utilisateurs ont des applications bien déterminées qui n'utilisent qu'une faible partie des ressources. Les premiers systèmes d'exploitation ont donc été prévus pour optimiser l'emploi de l'ordinateur et de ses périphériques : c'est la méthode du "traitement par lots". La plupart des utilisateurs s'estiment frustrés, surtout lorsqu'ils traitent des problèmes de recherche d'information ou de simulation : le temps écoulé entre la soumission du travail et la réponse est de quelques heures ou même d'un ou de plusieurs jours.

Pour de telles applications, un certain dialogue entre l'homme et la machine serait préférable. Mais cette communication pose un certain nombre de problèmes qui ont été étudiés, surtout à partir de 1950, par la US Navy, puis par le MIT et plusieurs universités américaines.

Nous allons préciser les différents problèmes et examinerons ensuite les différentes solutions possibles et la configuration utilisée pour notre travail de recherche.

1. Définitions

Les facteurs humains représentent tous les aspects de la communication entre l'homme et la machine. En effet, il ne faut jamais oublier que l'utilisateur est un être humain. Le système doit donc être étudié pour donner un maximum de confort pour accroître l'efficacité en gardant un bon temps de réponse et en éliminant les sources d'erreur.

2. Système interactif et programme interactif

Il existe deux degrés de communications possibles : le monologue et le dialogue.

Dans le premier cas, l'utilisateur formule son problème à l'aide d'un ensemble de commandes et doit attendre la fin de l'exécution (qui peut être très longue) avant de connaître les résultats et de pouvoir donner d'autres instructions.

Dans le deuxième cas, l'utilisateur peut être tenu au courant du déroulement de l'exécution et le modifier ou même l'interrompre, s'il le désire. Cette dernière condition est impossible à satisfaire à l'aide d'un programme interactif, écrit dans un langage du type FORTRAN standard.

Il existe en effet deux méthodes pour couper l'exécution : une interruption ou la consultation d'un "status". Dans les deux cas il est nécessaire d'écrire une routine en langage assembleur, effectuant une requête au système pour la sélection d'interruption ou la vérification du "status".

La première solution est plus rapide : le débranchement à la routine de traitement est immédiate, alors que le status ne sera lu qu'à des instants bien déterminés.

Le programme perd sa portabilité d'une installation à l'autre (voir chapitre V).

Le PL-1 et certains compilateurs FORTRAN temps réel fournissent une solution élégante au problème. Il est possible de définir des "tâches" avec des priorités différentes. Il suffit alors d'attribuer à l'entrée des commandes sur le terminal une priorité supérieure à celle de l'exécution d'une interrogation de fichier, par exemple. La recherche sera ainsi automatiquement interrompue lors de la frappe d'un caractère.

3. Terminaux interactifs

Le dialogue homme-machine requiert un certain nombre de dispositifs permettant de présenter l'information de manière compréhensible et capables d'en recevoir. La caractéristique principale de ces périphériques est leur vitesse de lecture/écriture. Mais nous verrons que d'autres éléments interviennent si on tient compte des facteurs humains comme la fatigue, l'inattention, etc.

Nous décrirons les différents types de terminaux et plus particulièrement notre terminal expérimental.

3.1. Machines à écrire

Ce type de console est le plus largement répandu. Les machines à écrire ont de nombreux avantages : leur fonctionnement est sûr, leur utilisation est simple, on obtient directement un journal imprimé des différentes transactions effectuées pendant la session interactive. Elles permettent d'ailleurs, depuis la première génération d'ordinateurs, le contrôle de la machine par l'opérateur. Enfin, leur coût est très faible.

Malheureusement, elles ont plusieurs inconvénients, en particulier leur vitesse de transmission (15 caractères/sec.) qui limite le volume de données qu'on peut imprimer; enfin elles sont mal adaptées aux applications graphiques [4].

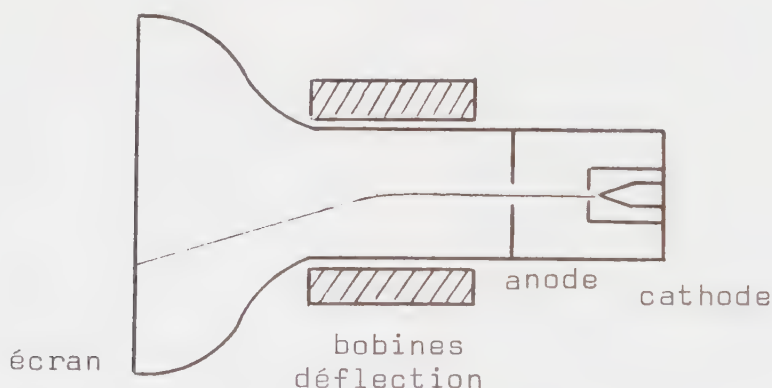
Les modèles les plus connus sont les Télétypes ASR 33 ou 35 et les modèles de bureau conventionnels : IBM 2740 ou 731.

3.2. Ecrans de visualisation

Les tubes à rayon cathodique, dérivés de l'oscilloscope, ne sont utilisés sur les ordinateurs que depuis 1952 environ [5]. Ils sont beaucoup plus rapides et souples que les dispositifs imprimants.

On en distingue un certain nombre de types suivant le genre de mémoire et suivant l'application : console pour un utilisateur ou grand écran.

Le principe du tube à rayon cathodique est simple : un faisceau d'électrons issu d'une cathode chauffée est accéléré et focalisé par un champ électromagnétique. Lorsqu'il atteint la face arrière de l'écran, revêtue d'une fine couche de "phosphore" (ZnS : Ag), il y a émission de lumière. Le faisceau peut être dévié par un champ électrostatique (plaques) ou un champ électromagnétique (bobines).



Comme il n'y a aucun mouvement mécanique, le tracé des symboles est extrêmement rapide. Mais en général, la persistance lumineuse est faible (1 $\mu\text{sec.}$ à 1/10 sec.) et l'image doit être mémorisée et régénérée assez souvent (30 fois par sec.) pour paraître stable.

Si l'excitation est brève et répétée, la luminance augmente jusqu'à un palier, puis oscille autour de cette valeur moyenne : c'est le phénomène de scintillement ou "flicker".

Certains paramètres doivent être fixés en fonction des facteurs humains : la luminosité et le contraste, la résolution, la lisibilité; d'autres dépendent essentiellement des applications : dimensions de l'écran, type de symboles, temps de réponse [6]. La luminosité doit être réglable en fonction de la lumière ambiante. Le contraste, défini comme le rapport entre la luminosité des symboles et celle du fond, ne doit pas être trop important (éblouissement). Le réglage optimum est différent suivant la couleur du fond : un contraste plus grand est nécessaire pour des caractères noirs sur fond blanc.

La résolution est une mesure de la discrimination spatiale et indique la dimension des détails reconnus comme distincts par l'oeil de l'utilisateur. Elle dépend essentiellement de la dimension du faisceau, de la granularité et de la diffusion du "phosphore".

On remarque ici deux exigences opposées : pour que l'écran soit lumineux, le faisceau doit être intense, mais alors le spot sera plus grand : le diamètre du faisceau augmente (phénomène de charge d'espace) et la diffusion dans les "phosphores" croît.

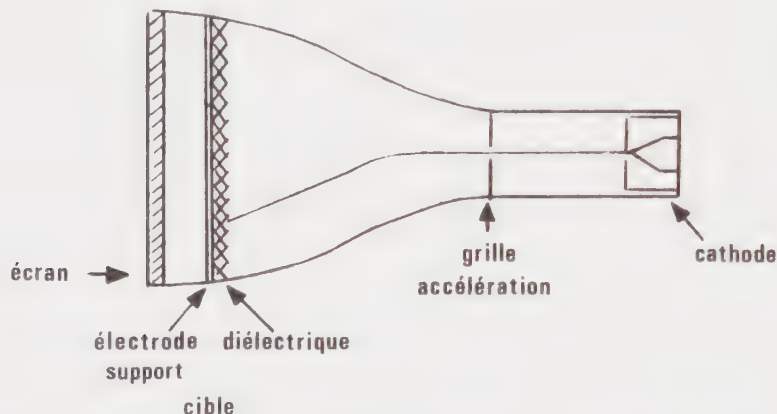
La dimension de l'écran, celle des caractères, leur forme, le rapport hauteur/largeur de symbole, la distance entre deux lignes affectent la lisibilité. Enfin la non-linéarité du balayage, la distorsion (en coussin ou en tonneau), relativement peu gênantes lors de l'affichage de textes, doivent être évitées dans les applications graphiques.

D'autres caractéristiques améliorent le confort de l'utilisateur et ne doivent pas être négligées : position et inclinaison de l'écran, filtres pour éviter les réflexions et améliorer le contraste [7].

Le stockage et la régénération de l'image sont réalisés suivant différents principes.

- Mémoire analogique.

La plupart des tubes-mémoire utilisent le principe de l'émission secondaire et d'accumulation de charges électriques sur des cibles diélectriques [8].



Ces tubes ont trois régimes de fonctionnement : effacement, écriture et lecture. L'écriture de la cible est obtenue par un faisceau d'électrons primaires issu d'une cathode chauffée, accéléré et focalisé. Lorsqu'il frappe la cible diélectrique, des électrons secondaires sont émis.

Si le rapport δ = nombre d'électrons secondaires émis/nombre d'électrons primaires incidents est supérieur à 1, il y a apparition de charges positives. Dans le cas contraire, il y a dépôt de charges négatives. Ce rapport δ varie avec la tension entre cathode et cible.

La lecture est simple : le faisceau d'électrons incidents peut traverser le diélectrique là où il a été chargé positivement par l'écriture, il peut alors atteindre l'électrode support ou l'écran. Au contraire, aux endroits chargés négativement, les électrons sont repoussés.

Remarque : plus la vitesse des électrons du faisceau de lecture est faible, plus le nombre d'électrons secondaires produit sera petit; l'image sera visible plus longtemps mais avec un contraste moindre.

Le type le plus connu est l'écran à vision directe (Tektronix 611). La persistance est de plusieurs minutes, ce qui élimine pratiquement la nécessité de "rafraîchir" l'image. Mais l'effacement n'est possible que pour une page entière.

- Mémoire digitale.

Il faut enregistrer l'image point par point à moins que des générateurs de vecteurs (graphiques) ou de caractères ne soient disponibles. Dans ce cas, les caractères sont stockés sur une mémoire auxiliaire à accès relativement rapide : tambour, ferrites ou semiconducteurs MOS. A partir de cette information binaire codée, les symboles sont formés par le générateur de caractères et un convertisseur digital-analogue commande les tensions (ou les courants) nécessaires à la déviation du spot.

3.3. Claviers

Il existe deux catégories de consoles pour l'entrée des données : le clavier alphanumérique et le clavier de fonctions.

Le premier permet la formation classique des commandes caractère par caractère. La frappe est lente et les erreurs peuvent être nombreuses, les chercheurs n'étant en général que de médiocres dactylographes. Dans la conception ou dans le choix d'un tel clavier, deux conditions sont à respecter : éviter au maximum dans les applications envisagées, la manipulation du contrôle minuscule/majuscule et prévoir dans le cas d'une entrée ligne par ligne une mémoire tampon suffisamment rapide : l'utilisateur pourra ainsi porter toute son attention sur sa recherche, sans contrôler un indicateur "TYPE IN".

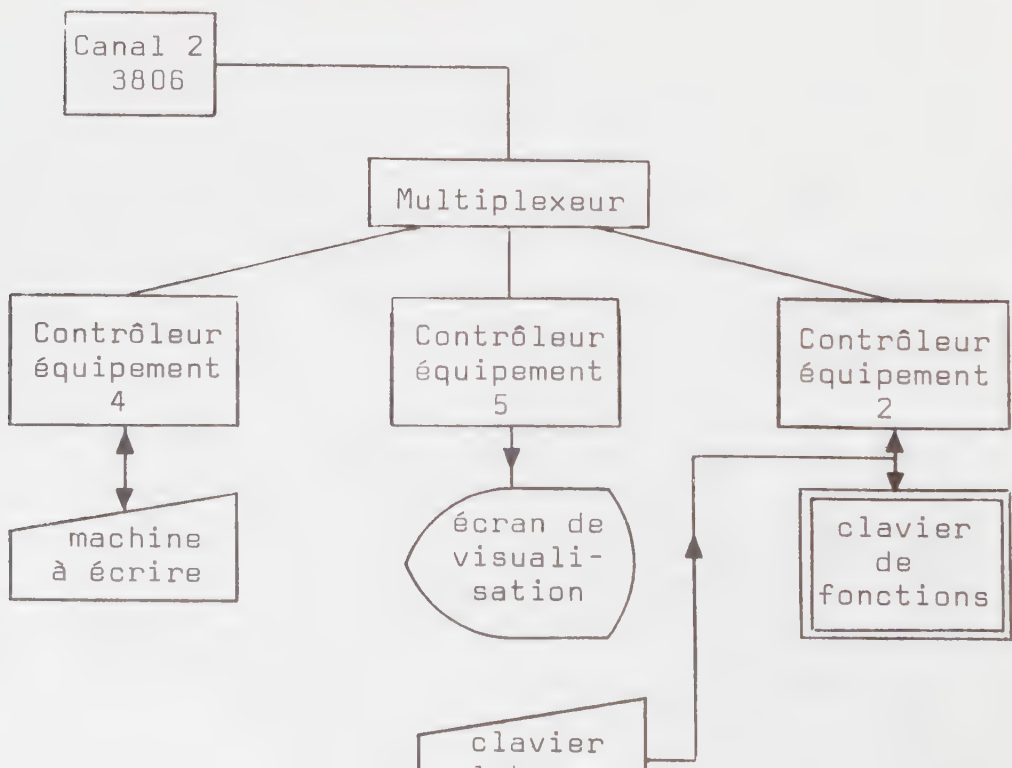
Le clavier de fonctions est beaucoup plus évolué : il comporte un ensemble d'interrupteurs ou de boutons poussoirs auxquels sont associées des fonctions complètes : on élimine ainsi les erreurs de frappe. D'autre part, des témoins lumineux peuvent indiquer les commandes admises à un instant donné (voir chapitre IV.7.). WEAR a étudié les per-

formances de ce système [9] : après un apprentissage de courte durée, on observe un gain de temps de 23 % par rapport à la méthode traditionnelle.

Mentionnons encore le crayon "lumineux". Il est constitué d'un détecteur photoélectrique. L'écran est balayé selon un standard télévision et la cellule associée à un circuit logique repère la position du crayon sur l'écran. Cette méthode est utilisée dans deux applications : le "tracé" de dessins sur l'écran où l'image est digitalisée point par point et la sélection de fonctions. Le programme indique les options possibles et le crayon permet de choisir avec un risque d'erreur extrêmement faible.

3.4. Configuration utilisée

Nos premières expériences ont été réalisées sur un ordinateur de deuxième génération CDC 3800. Le terminal a été étudié par un groupe du Centre Interfacultaire de Calcul Electronique et connecté par un multiplexeur à l'un des quatre canaux 3806 [10]. Il est composé de trois équipements commandés chacun par un contrôleur.



La machine à écrire est un modèle de bureau classique à commande électromagnétique IBM 731. Un ruban bicolore permet de distinguer a posteriori le texte imprimé par l'ordinateur (rouge) et celui tapé par l'utilisateur (noir). Deux caractères spéciaux permettent l'annulation d'un caractère (@) ou d'une ligne (φ). La vitesse maximum d'écriture est de 16 caractères/seconde; un bouton-poussoir permet de couper une sortie trop longue.

L'écran de visualisation utilise un alphécon (tube à mémoire électrostatique) et un moniteur de télévision : ce système permet l'affichage simultané sur plusieurs récepteurs. La taille des caractères peut être choisie en fonction de l'application. La limite de lisibilité est atteinte avec 46 lignes de 80 caractères, mais les conditions d'utilisation normales sont de 35 lignes de 60 caractères.

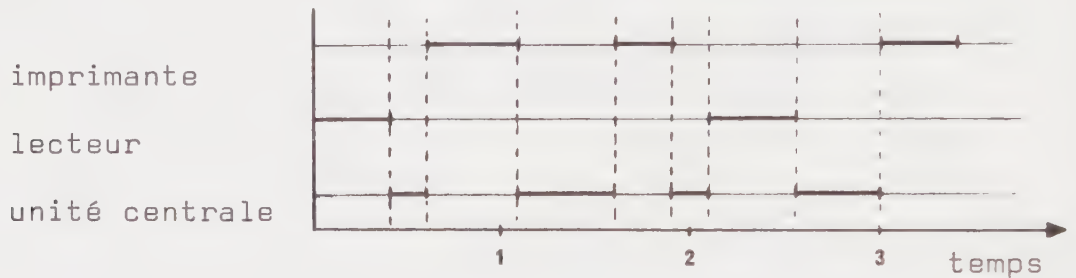
Le clavier alphanumérique est un Honeywell 67SW1-2 à touches à effet Hall. Un certain nombre de caractères spéciaux sont utilisés par une routine de système de liaison clavier écran pour déterminer la taille des caractères, effacer l'écran. Cette routine est aussi nécessaire pour l'affichage simultané des caractères et des fonctions, car il n'y a aucun lien physique.

Le clavier de fonctions est constitué d'une grille de faisceaux lumineux. Le doigt ou n'importe quel objet opaque coupant deux de ces rayons sélectionne une fonction. Un ensemble de 64 lampes associées, placées sous la grille, désigne à chaque instant les options disponibles.

Tous ces équipements sont reconnus par le système d'exploitation de la CDC 3800 : DISK SCOPE. Ils sont accessibles directement à l'aide des ordres READ/WRITE du FORTRAN, soit en BCD pour les données, soit en binaire pour les fonctions, comme la sélection de taille ou d'intensité.

4. Systèmes d'exploitation

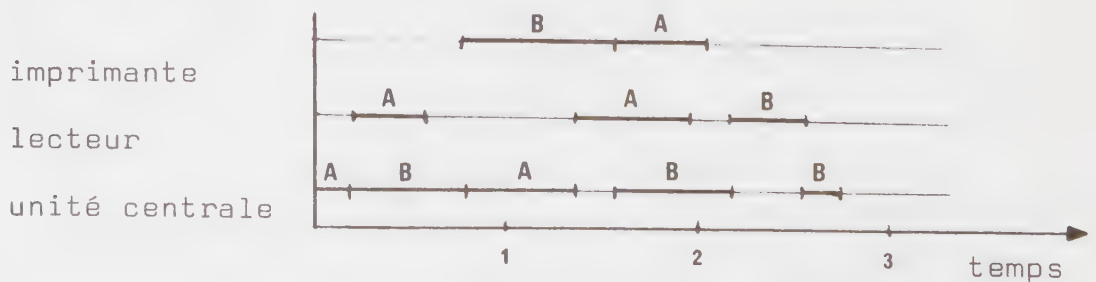
Les premiers ordinateurs étaient monoprogrammés : un seul travail est exécuté à la fois. L'utilisation des ressources est en général assez faible : plusieurs périphériques restent totalement inactifs, la mémoire est partiellement remplie. De plus les éléments rapides, unité centrale et mémoires, doivent souvent attendre la fin d'une opération d'entrée-sortie pour pouvoir continuer. Le rendement est mauvais.



Dans l'exemple ci-dessus, les divers éléments travaillent :

imprimante	6/17	35%	du temps
lecteur	4,5/17	27%	" "
unité calcul	6,5/17	38%	" "

En multiprogrammation, plusieurs travaux se déroulent simultanément. Dans la version la plus simple, un programme garde le contrôle jusqu'à ce qu'une opération de lecture ou d'écriture soit nécessaire. Un autre programme est alors traité en mémoire centrale.



Dans l'exemple ci-dessus, deux travaux A et B partagent les ressources disponibles. Le taux d'utilisation est meilleur.

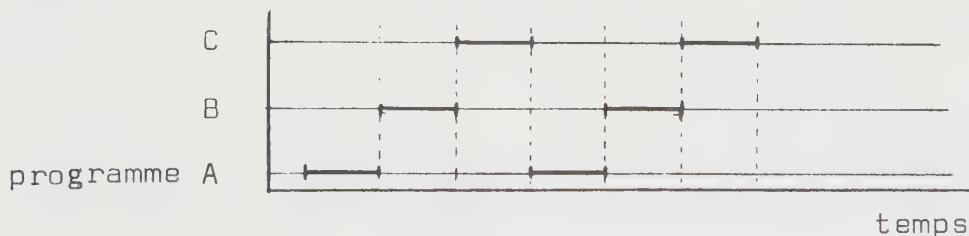
imprimante	6,5/14	46%	du	temps
lecteur	7/14	50%	"	"
unité centrale	11/14	71%	"	"

Il est possible d'augmenter le rendement en traitant simultanément un nombre de programmes supérieur.

Ce système présente un grave inconvénient : un travail peu prioritaire, mais n'effectuant aucune opération d'entrée-sortie, peut garder le contrôle.

Une horloge générant des interruptions à intervalle régulier permet de résoudre ce problème : toutes les 10 ou 100 msec. le programme présent en unité centrale doit céder le contrôle à un autre : un algorithme doit déterminer lequel, en tenant compte des priorités et des entrées/sorties en cours de traitement.

En temps partagé, on cherche à offrir à chaque utilisateur l'ensemble des ressources hardware et software au même instant macroscopique : chacun aura l'impression que l'ordinateur ne travaille que pour lui. Des tranches de temps égales sont attribuées cycliquement aux divers utilisateurs.



Plusieurs caractéristiques sont importantes :

- nombre de terminaux pouvant travailler simultanément,
- partage des ressources,
- temps de réponse.

Le partage des ressources pose un certain nombre de problèmes :

- La mémoire n'est jamais assez grande pour contenir tous les programmes simultanément : il est nécessaire d'échanger les programmes entre une mémoire auxiliaire et la mémoire centrale (swapping). Mais dans le cas général, le rechargement ne s'effectue pas à la même adresse; il faut donc une translation automatique des adresses à l'exécution.
- Certains sous-programmes sont appelés par plusieurs utilisateurs (routines de bibliothèque, éléments de compilateur); une copie différente pour chacun serait coûteuse, il est nécessaire que ces sous-routines soient écrites de manière réentrante.

Ces problèmes seront examinés lors de l'étude des performances de notre interpréteur (chapitre VI).

Enfin le partage des ressources implique un risque de transfert non souhaité, soit fortuit : erreur dans le système, soit délibéré : espionnage.

Malgré les problèmes indiqués, un programme interactif n'est utilisable que sur une machine multiprogrammée (rentabilité). La technique de time-sharing n'est pas indispensable, mais améliore considérablement le temps de réponse.

Chapitre II : Structures de données

La technique utilisée pour l'enregistrement des données détermine souvent les performances d'un système interactif. Nous allons montrer que le choix d'une méthode ou d'une autre dépend non seulement de facteurs économiques (périphériques à disposition, par exemple) mais aussi des applications envisagées.

1. Définitions

Avant d'examiner les différentes techniques d'organisation de fichiers, il est important de définir certains termes : structure de données, structure de fichier, base de données, etc. Ces définitions correspondent à celles du COBOL Standard USASI [11, 12, 13].

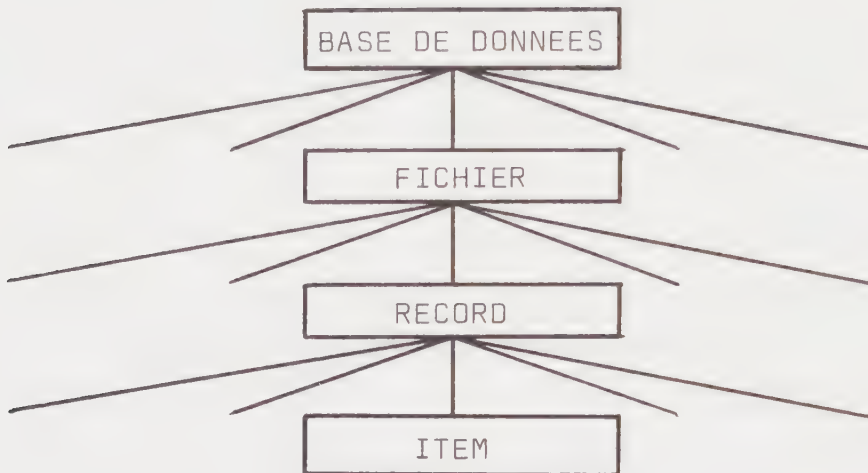
Un ensemble de données peut être examiné selon deux points de vue différents :

- logique : la structure d'information est déterminée par le programmeur d'applications;
- physique : l'organisation du fichier est imposée par l'analyste en fonction de plusieurs paramètres, en particulier le temps de réponse, la logique de recherche, les périphériques utilisés, etc.

Une base de données est un groupe de fichiers ou, sans considérer l'organisation en fichiers, un ensemble de données logiquement reliées et accessibles par un ou plusieurs programmes. Une des caractéristiques nécessaires est l'indépendance des données vis-à-vis du programme. Pour satisfaire cette exigence, la définition des données doit être enregistrée avec l'information.

La manipulation de données ou "data management" désigne le mouvement des données de et vers la mémoire. Dans un sens plus large, c'est l'ensemble des procédures qui permettent de définir puis de maintenir l'intégrité et la sécurité des données.

L'organisation verticale d'une base de données est la suivante [14] :



Un fichier est un agrégat arbitraire, mais pas forcément homogène, d'enregistrements.

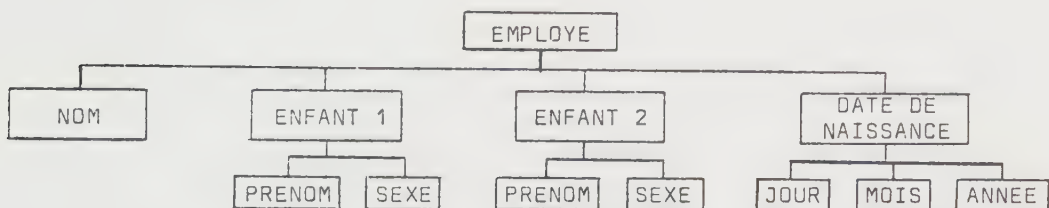
Un enregistrement ou "record" est un ensemble d'une ou de plusieurs données élémentaires ou "items", auxquelles sont associées un nom et une valeur (paire attribut-valeur).

La base de données d'une entreprise, par exemple, contiendra plusieurs fichiers : personnel, clients, stocks, etc.

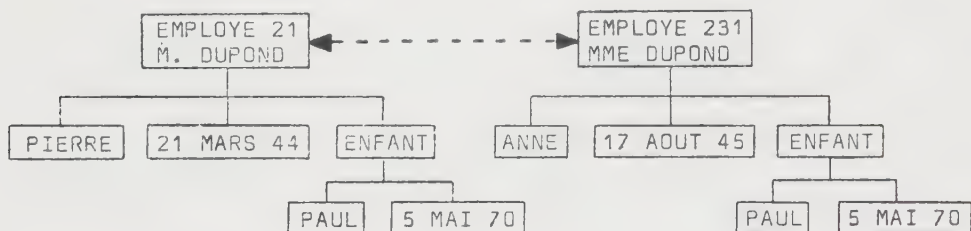
Dans le fichier personnel, l'ensemble des renseignements correspondant à un employé constitue un record dont les différents items sont le nom, l'adresse, l'état civil, etc.

Un groupe est un ensemble d'items ou d'autres groupes. Il est appelé simple dans le premier cas et composé dans le deuxième cas. On peut diviser, par exemple, le groupe DATE en trois parties : JOUR, MOIS, ANNEE et référencer ensuite aussi bien DATE que JOUR, MOIS, ANNEE.

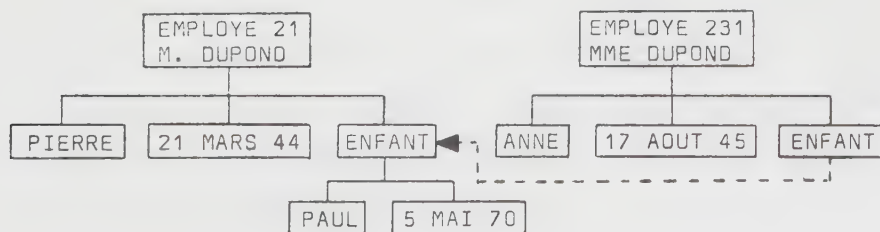
Certains groupes se répètent, d'autres non : un employé peut avoir plusieurs enfants mais n'a qu'une seule date de naissance. Les structures arborescentes (voir paragraphe 5) sont spécialement bien adaptées pour représenter de telles relations.



Il existe certaines relations entre les groupes dans le sens vertical (relations hiérarchiques) et horizontal (relations associatives).



On remarque ici un cas extrêmement fréquent : la redondance d'information; il est possible de remplacer le deuxième stockage de données par un jeu de pointeurs vers le premier.



Nous allons examiner maintenant les principales caractéristiques des organisations de fichiers les plus souvent utilisés [15].

2. Organisation séquentielle

L'organisation séquentielle des fichiers est la plus ancienne méthode et certainement la mieux connue. Les enregistrements sont placés les uns à la suite des autres, suivant un certain ordre qui est souvent celui d'arrivée dans le fichier. L'ordre physique et l'ordre logique sont identiques.

L'organisation est simple et lorsqu'on traite l'enregistrement n , l'accès à l'enregistrement $n+1$ est très rapide. Par contre l'accès à un endroit quelconque d'un fichier non trié est lent : si N est le nombre total d'enregistrements dans le fichier, le temps est proportionnel à $N/2$.

$$t_n \div \frac{N}{2}$$

En effet, il faut parcourir le fichier séquentiellement. Pour une recherche rapide, il est intéressant de l'avoir trié selon la clé d'accès la plus fréquente; mais alors plusieurs copies sont nécessaires si on désire interroger suivant des clés différentes.

La mise à jour est lente : il faut copier le fichier complet, mais l'insertion, la suppression ou la modification d'un enregistrement est rapide surtout si les champs sont de longueur fixe.

Le principal avantage de l'organisation séquentielle est donc l'accès rapide à des enregistrements consécutifs; mais il disparaît lorsque le fichier est stocké sur un disque dans un environnement de multiprogrammation : la position du bras change pour chaque utilisateur et les "records" n'apparaissent plus consécutivement.

Par contre sur un périphérique à accès direct, une autre possibilité existe : la recherche binaire.

Il faut d'abord déterminer la clé principale d'accès au fichier, c'est-à-dire l'élément d'information qui sera le plus souvent utilisé lors d'une recherche. Le fichier sera trié, à la création, selon ce champ.

Exemple :

	46	DUPOND	ETUDIANT	
	151	DUPONT	MECANICIEN	
	29	DURAND	ELECTRICIEN	
	37	MARTIN	MENUISIER	
	205	TERRIER	MEDECIN	
Numéro		Nom	Profession	

On cherche les renseignements concernant Monsieur MARTIN. Suivant la méthode séquentielle, il est nécessaire de parcourir 4 "records".

En recherche binaire, on procède par dichotomie : on examine l'élément central DURAND. Ce nom est "inférieur" dans l'ordre alphabétique à celui qui nous intéresse. On prend alors l'élément médian de la deuxième partie du fichier : MARTIN. La recherche est terminée.

En moyenne, le temps nécessaire est :

$$t_n \div \log_2 N$$

pour un fichier contenant N éléments.

3. Organisation "random"

Cette méthode a été développée pour éviter les inconvénients mentionnés ci-dessus, lorsqu'on désire accéder à un enregistrement quelconque. Les "records" sont placés dans le fichier à une position déterminée en appliquant une certaine relation à la clé. Cette adresse doit être calculée non seulement à la création du fichier, mais aussi lors de chaque recherche.

Il existe trois types d'organisation "aléatoire" :

- adresse directe,
- recherche dans une table,
- calcul.

3.1. Adresse directe

L'adresse "hardware" est donnée par la valeur de la clé. Cette méthode est très rapide, mais les valeurs doivent être uniques et suffisamment significatives pour l'utilisateur, puisque la recherche doit s'effectuer à l'aide de ces adresses.

3.2. Recherche en table

La clé et l'adresse physique de l'enregistrement sont placés dans une matrice à deux dimensions :

entrée →	rabot	175
	marteau	3
	scie	27
	tenailles	41
	clé	adresse

Les opérations sont simples : pour ajouter un enregistrement, on ajoute une entrée, pour supprimer un "record", on efface l'entrée correspondante.

La recherche dans la table peut s'effectuer séquentiellement. Le temps t_1 nécessaire pour trouver l'adresse est proportion-

nel à $\frac{N+1}{2}$ si N est le nombre d'entrées dans la table.

Si la table est triée suivant l'ordre croissant ou décroissant des clés, une recherche binaire est possible; le temps t est alors proportionnel à $\log_2 N$. L'adresse est trouvée beaucoup plus rapidement, mais la mise à jour du fichier nécessite la réorganisation complète de la table.

Le temps d'accès à un enregistrement quelconque est donc :

$$t = t_1 \text{ (recherche dans la table)} + t_2 \text{ (accès périphérique)}$$

t_1 est peu important si la table est suffisamment petite pour être stockée en mémoire centrale.

3.3. Calcul

Un algorithme permet de transformer la clé en une adresse. Comme le nombre de combinaisons possibles des caractères de la clé est plus grand que la place disponible, l'algorithme devra exercer une certaine compression.

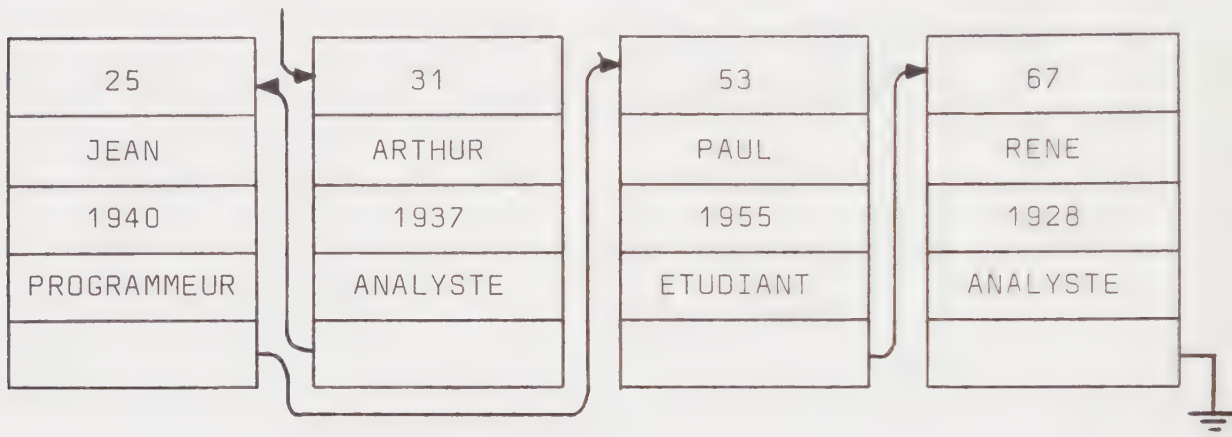
Deux méthodes sont possibles : compression et "hashing".

- Compression : on supprime de la clé les caractères les moins significatifs. Cette technique est particulièrement utile lorsque les données sont des noms qui peuvent être transcrits phonétiquement. L'algorithme élimine alors les lettres muettes, réduit les caractères doubles, etc.
- "Hashing" : on applique une fonction f à une clé pour obtenir une adresse. Plusieurs fonctions différentes ont été mises au point :
 - on attribue à chaque caractère un code numérique et on additionne tous ces nombres pour obtenir l'adresse;
 - on peut aussi considérer la clé comme un nombre, l'élever au carré et prendre les bits du milieu comme adresse.

Mais pour deux clés distinctes, les adresses calculées ne sont pas toujours différentes : il se produit des "collisions". Il faut alors chercher dans les positions suivantes une place libre ou consulter une table ("bump table"). Le traitement de ces collisions détermine la rapidité de la méthode.

4. Structure de liste

Dans une organisation séquentielle, l'ordre logique et l'ordre physique des enregistrements sont les mêmes. L'utilisation de pointeurs permet de séparer complètement ces deux notions. Dans chaque enregistrement, un champ contient l'adresse de l'enregistrement logique suivant.



Dans cet exemple, les enregistrements sont triés logiquement par ordre alphabétique, mais organisés physiquement selon la séquence d'entrée dans le fichier. La convention utilisée pour la représentation de la fin de liste (symbole "terre") est celle adoptée par Knuth [16].

4.1. Liste simple (chaîne)

Chaque enregistrement n'a qu'un seul prédécesseur et un seul successeur; la suppression ou l'insertion d'un "record" est très facile, il suffit de modifier un pointeur. Certaines listes n'admettent les modifications qu'à une extrémité, ce sont les piles ou "stacks" et les queues.

4.2. Fichier inversé

Un fichier inversé est constitué de listes inversées. Chacune de celles-ci est une chaîne dont tous les enregistrements ont la même valeur pour une clé donnée.

En reprenant l'exemple précédent, les "records" 31 et 67 appartiennent à la même liste inversée si la clé choisie est la profession. Il existe donc une relation associative entre ces deux enregistrements.

ANALYSTE	31	67	
ETUDIANT	53		
PROGRAMMEUR	25		

En réalité, on ne trouve pas directement le numéro de l'enregistrement, mais son adresse.

Un fichier inversé permet un accès rapide à tous les enregistrements dont la valeur d'une clé est identique. Le fichier peut être soit partiellement inversé, soit totalement inversé. Ce dernier cas est très favorable pendant les interrogations, mais très défavorable lors de la mise à jour : toutes les listes inversées doivent être corrigées.

4.3. Anneaux

Un anneau est une liste linéaire qui boucle sur elle-même. La recherche peut commencer en n'importe quel point de la liste, mais l'algorithme de recherche doit mémoriser le point d'entrée pour éviter une boucle sans fin. La recherche dans un anneau étant séquentielle est beaucoup plus lente, surtout si les anneaux sont longs.

Toutes les méthodes utilisant des listes nécessitent une place en mémoire plus importante à cause des pointeurs.

5. Structure arborescente

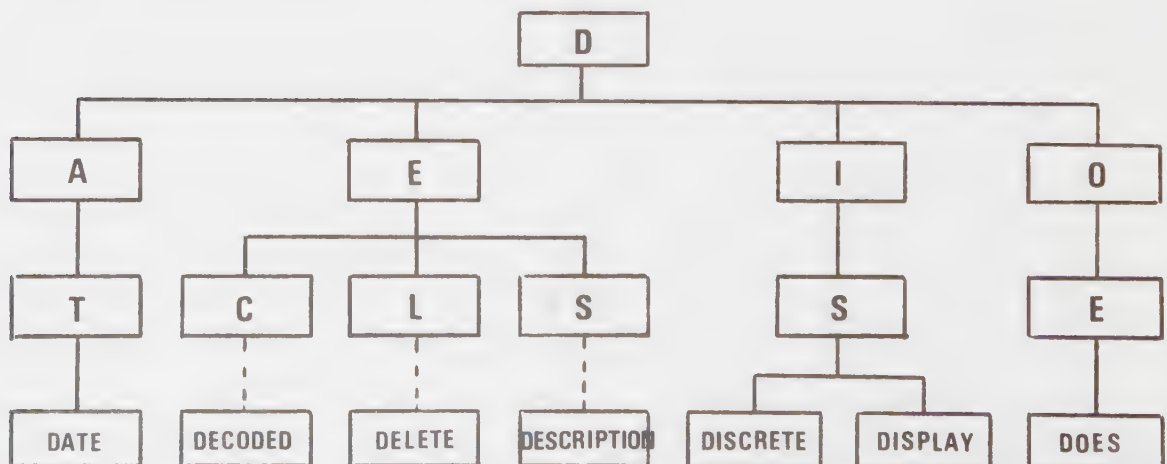
Les tables d'accès aux enregistrements sont organisées de manière hiérarchique. L'arbre possède une racine par laquelle commence la recherche. Cette racine donne l'adresse des noeuds du niveau 1 (clé principale). Lorsqu'on a trouvé le noeud correspondant à la valeur désirée, celui-ci contient un pointeur vers certains noeuds du niveau 2. L'arbre est parcouru ainsi jusqu'au niveau inférieur qui possède les pointeurs vers l'information.

Le problème principal est celui de la traversée de l'arbre en un temps minimum. De nombreux algorithmes ont été mis au point (voir en particulier [16]).

La méthode de recherche binaire ou recherche "intelligente" exposée au paragraphe 2 est très semblable. La racine de l'arbre binaire serait constituée par la valeur centrale, la branche gauche représenterait les valeurs inférieures et la branche droite les valeurs supérieures; cette structure est identique à tous les niveaux.

Exemple : Recherche d'un mot-clé dans un dictionnaire.

Le premier niveau sélectionne la première lettre du mot, le second la deuxième lettre et ainsi de suite jusqu'à la "feuille"; on a alors parcouru le mot en entier.



6. Organisation séquentielle indexée

Les enregistrements sont placés séquentiellement dans le fichier trié selon une clé, mais un index permet d'accéder à un "record" quelconque.

Exemple :

Un fichier contenant les informations sur le personnel d'une entreprise sera trié selon l'ordre alphabétique. Si lors d'une recherche, un directeur veut la liste des employés d'une division donnée, l'index correspondant permettra d'accéder directement aux différents enregistrements. De plus ceux-ci apparaîtront déjà en ordre alphabétique (clé de tri principale).

La recherche dans un index est beaucoup plus rapide que le parcours séquentiel du fichier. On sera donc amené à utiliser les techniques d'accès aléatoire.

Lorsqu'on est positionné sur un enregistrement, le passage au suivant nécessite la répétition de la procédure complète. D'autre part, les pointeurs occupent une place non négligeable en mémoire. Enfin, si l'index est important, il devra être stocké sur disque et l'accès à un enregistrement nécessitera un double accès au périphérique.

La mise à jour est très simple car seuls les "records" modifiés sont réécrits et les suppressions ne sont pas physiquement effectuées; l'entrée dans l'index est effacée et on utilise ensuite une technique de "ramassage d'ordures".

Chapitre III : Description de l'interpréteur INFOL

INFOL, acronyme de INformation Oriented Language, est un système de traitement de fichiers développé par T.W. Olle chez Control Data; il est opérationnel depuis 1966 sur les ordinateurs CDC de la série 3000 Upper [17].

INFOL était destiné essentiellement à permettre à des chercheurs sans expérience informatique de traiter des fichiers de taille modeste. C'est un système "auto-contenu", appelé aussi procédural.

Nous préciserons ces notions et indiquerons ensuite la structure de données utilisée et les fonctions principales du système.

1. Définitions

Il est possible de distinguer deux catégories parmi les nombreux systèmes de manipulation de données disponibles aujourd'hui.

La première correspond à des extensions d'un langage de base, par exemple le COBOL ou le PL/1, permettant de traiter des structures plus complexes.

Exemple :

DBTG : Data Base Task Group Proposal (CODASYL, 1971).

IDS : Integrated Data Store (Honeywell, 1963).

La deuxième catégorie désigne les systèmes où les algorithmes de traitement sont déjà programmés (systèmes "auto-contenus"). L'utilisateur ne peut intervenir dans le déroulement des opérations. En général, les applications traitées par ce type de langage seront de dimension moindre, mais nécessitent un temps de programmation et de mise au point bien inférieur.

Exemple :

GIS : Generalized Information System (IBM, 1969).

TDMS : Time-shared Data Management System (SDC, 1969).

2. Définition et organisation des données

La structure des données est hiérarchique avec plusieurs niveaux. Chaque enregistrement du fichier, comme il a été défini dans le chapitre II est appelé ici "élément".

Un élément contient lui-même un certain nombre de renseignements ou variables. Ce sont les "items".

On en distingue deux catégories : simple ou multiple. Le langage INFOL étant un sous-ensemble de l'anglais, nous utiliserons quelquefois la terminologie correspondante : les items peuvent être UNARY ou MULTIPLE.

Exemple :

L'item contenant le titre d'un ouvrage est simple. Au contraire, l'item correspondant aux auteurs (il peut y en avoir plusieurs pour une publication) sera multiple.

Dans l'exemple ci-dessus, chaque auteur est un sous-item.

Il y a donc trois niveaux : ELEMENT
ITEM
SOUS-ITEM.

Chaque item est identifié par un numéro et un nom (ou description). Pendant tout le traitement, les variables doivent être référencées par leur numéro qui doit être unique; la description n'est utilisée que lors de l'impression des résultats et le même nom peut apparaître plusieurs fois.

2.1. Définition du champ

Quatre types d'informations sont reconnues :

- alphanumérique : une chaîne de caractères de longueur quelconque (<250 symboles) mais ne contenant pas d'astérisques.

- numérique : nombres entiers ou réels dont la précision ne dépend que de l'ordinateur utilisé. INFOL n'étant pas spécialement destiné aux applications mathématiques, une limite supplémentaire est imposée : le nombre de chiffres composant la partie décimale doit être inférieur ou égal à 10.

En plus de ces deux types définis dans tous les langages, il existe deux classes spéciales :

- dates : sous forme complète JOUR/MOIS/ANNEE ou incomplète MOIS/ANNEE ou seulement ANNEE. Toutes les dates sont converties et stockées en format interne.
- codes : un dictionnaire permet de lier un mnémonique ou un numéro à un texte. Seule cette forme "réduite" est utilisée et enregistrée en mémoire.

2.2. Définition de l'enregistrement

Les renseignements à stocker peuvent être de longueur tout à fait quelconque et variable. Habituellement, on doit réserver pour chaque variable une place constante et au moins égale à la longueur maximale qu'on pourra rencontrer. Cette méthode, utilisée en COBOL par exemple, prendra, inutilement la plupart du temps, beaucoup de place.

Pour éviter cet inconvénient, INFOL utilise un système de pointeurs, placé en tête de chaque élément. Cet ensemble de pointeurs est appelé HEADER, dont le premier mot indique la longueur de l'élément.

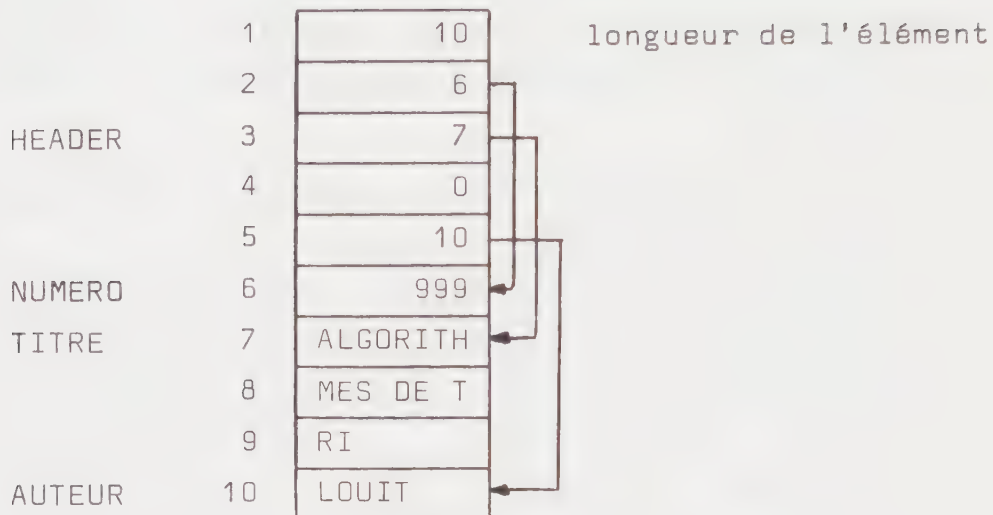
Le pointeur correspondant à un item donne l'adresse du début du champ relative au début de l'élément.

Remarque :

Cette méthode introduit une grande souplesse dans la structure de données, mais si l'information est de longueur constante, les pointeurs occupent inutilement une fraction de la mémoire.

Exemple :

On a quatre items par élément :



Remarque :

Lorsqu'un item n'a pas de valeur, son pointeur est nul.
Ceci est valable pour les items simples.

Pour les items multiples, l'adressage est indirect.

Le premier pointeur indique l'adresse du nombre de sous-items. Ceux-ci sont placés dans des positions consécutives avant ce nombre.

Deux cas peuvent se présenter :

- l'item est numérique : la valeur n'occupe qu'un mot-machine.
L'adresse est donnée directement par la relation :

$$IAD \text{ (sous-item NS)} = IAD \text{ (nb. de sous-items)} - NSUB + NS - 1$$

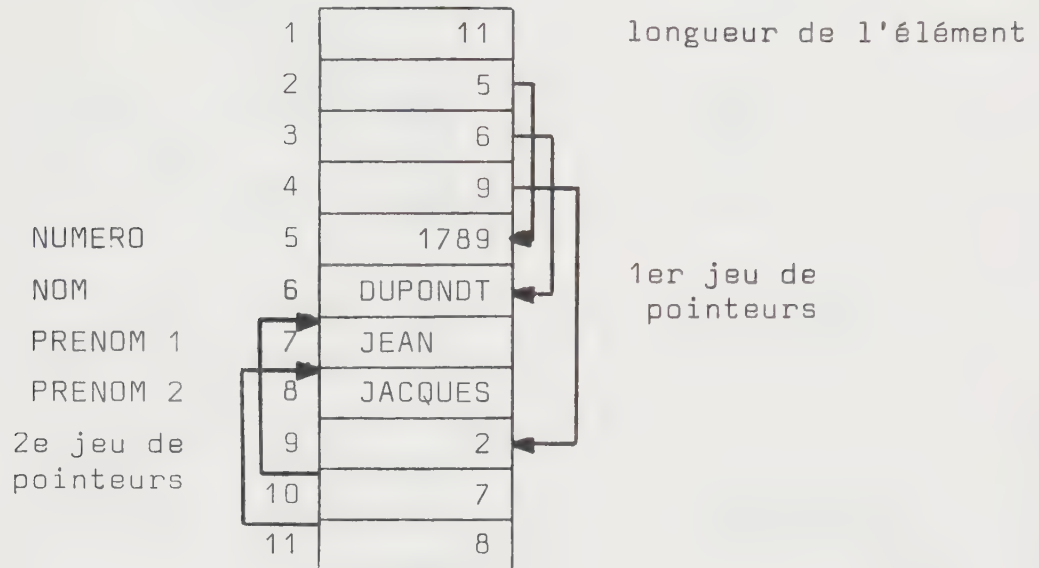
où NSUB = nombre de sous-items

NS = numéro du sous-item.

- l'item est alphanumérique : la chaîne de caractères est de longueur variable, un deuxième jeu de pointeurs est nécessaire. Chacun indique l'adresse du début du sous-item correspondant.

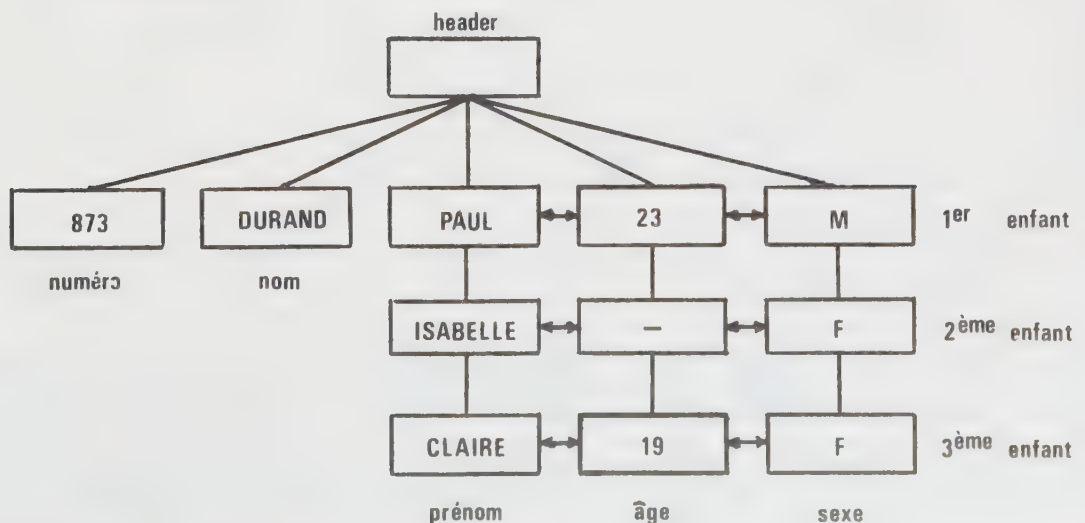
Exemple :

L'item 3 est multiple et contient 2 sous-items :



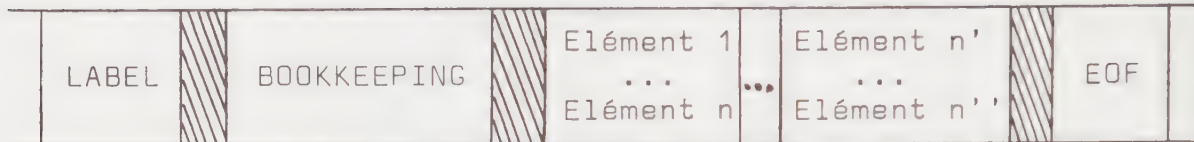
Il est possible d'associer certains items multiples, même de types différents pourvu qu'ils aient le même nombre de valeurs (sous-items). La notion de valeur manquante, représentée par le signe -, a été introduite pour satisfaire cette condition. Le traitement de ces valeurs manquantes est particulier : elles ne sont pas comptées lors des opérations arithmétiques par exemple. Le nombre d'items associés peut être quelconque.

Exemple :



2.3. Structure du fichier

L'organisation des données est séquentielle. Le fichier INFOL se présente de la manière suivante :



Le label contient le mot de contrôle INFOL, le nombre d'éléments présents dans le fichier, la date de création et de la dernière mise à jour et un mot de passe (optionnel) permettant de contrôler l'accès aux données.

Le bookkeeping contient un résumé des caractéristiques essentielles du fichier. Cet enregistrement, de longueur variable, donne à l'ordinateur les renseignements nécessaires au traitement du fichier :

- le nombre d'items;
- l'adresse de la description de chaque item. Cette description pourra être imprimée lors de l'édition.
- la catégorie et le type de chaque item;
- l'adresse des codes;
- les validations imposées : entier, alphabétique, valeurs limites;
- le nombre maximum de sous-items trouvés pour un item déterminé pendant la création du fichier;
- la longueur maximale d'un champ;
- les codes;
- les valeurs limites.

Les éléments sont placés séquentiellement à la suite du bookkeeping.

Pour des raisons de temps de lecture et d'accès aux unités de bande magnétique ou de disque, les éléments (enregistrement logique) seront groupés en blocs de longueur à peu près fixe (2000 mots).

Exemple :

élément 35
élément 36
élément 37

1980 mots

élément 38

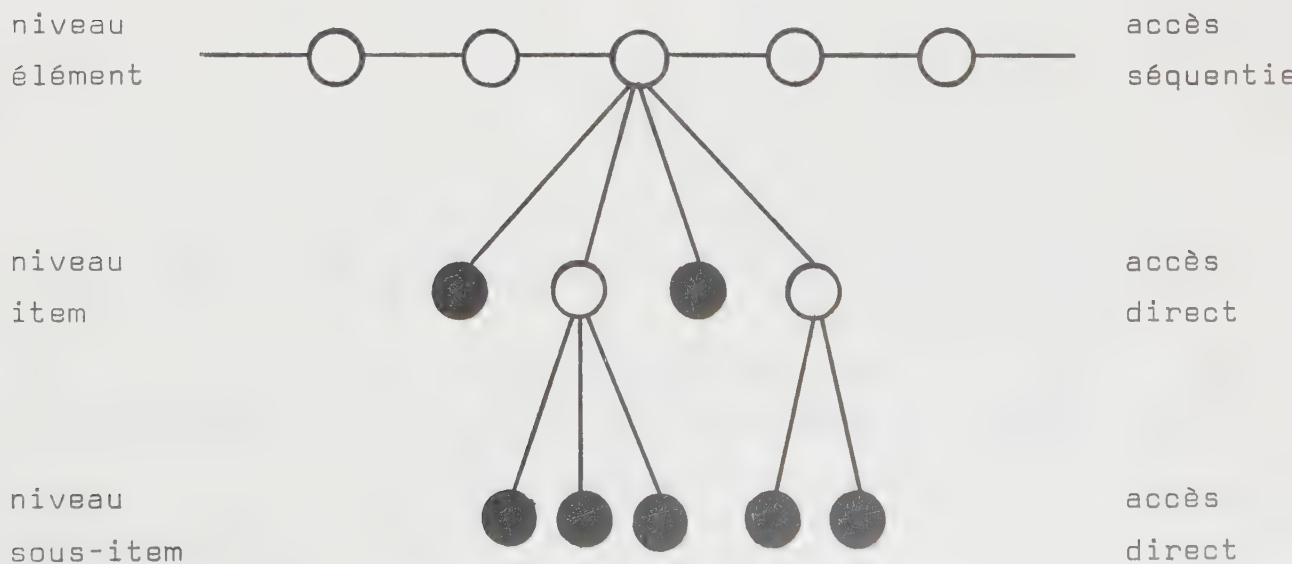
1700 mots

élément 39
élément 40

1850 mots

Le fichier est terminé par un end-of-file.

En résumé, l'accès aux différents éléments est séquentiel, mais grâce au système de pointeurs, l'accès à tous les niveaux inférieurs est direct.



Les cercles pleins représentent le niveau auquel on trouve les valeurs.

3. Fonctions principales du système

INFOL permet d'effectuer trois fonctions importantes : création, interrogation et mise à jour, mais il n'est capable de traiter qu'un seul fichier à un instant donné [18].

3.1. Création du fichier

Cette phase, commandée par le mot de contrôle ESTABLISHMENT, comporte en réalité deux sections bien distinctes : la définition des données et l'entrée des données (appelée INITIAL INPUT).

- 3.1.1. La définition n'est effectuée qu'une seule fois lors de la création du fichier. Elle précise :
- l'identification des items,
 - la structure (items simples ou multiples, associations),
 - les codes,
 - éventuellement les validations imposées.

Dans la version actuelle de l'interpréteur, il n'est pas possible de modifier ces définitions dans une autre phase.

- 3.1.2. La deuxième partie du traitement correspond à la conversion des données à partir d'un fichier source suivant les indications données dans la première partie.

Au moins un élément doit être présent pour que la création du fichier soit effective. Tous les éléments suivants peuvent être introduits lors d'une opération de mise à jour : le format d'entrée est exactement le même.

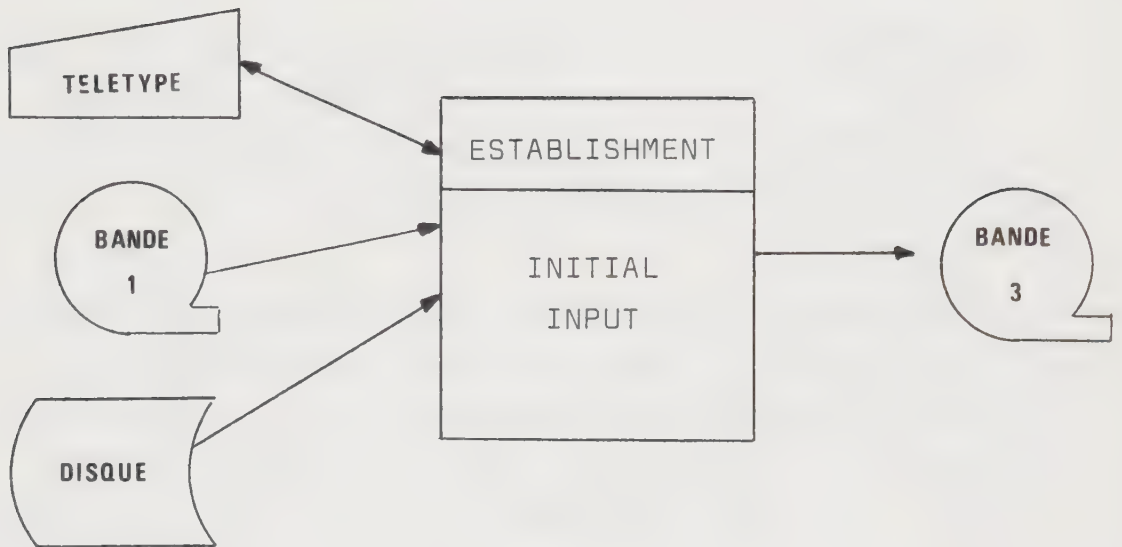
Chaque valeur doit être précédée du numéro d'item encadré d'astérisques.

Les sous-items sont séparés par des astérisques; un astérisque supplémentaire indique la fin d'un item multiple.

Reprenons l'exemple de la page

Les données devraient être fournies de la manière suivante :

```
*1* 1789    *2* DUPONT    *3* JEAN*JACQUES**
                                /      \
                           fin        fin
                        d'item    d'élé-
                        multiple   ment
```



En mode interactif, la définition est faite à partir d'un clavier alphanumérique ou d'un télétype. L'INITIAL INPUT peut traiter des données venant soit du terminal, soit d'un fichier source existant sur bande magnétique, disque, etc. Le clavier ne sert alors qu'à la correction des erreurs détectées. Lorsque le fichier est peuplé, des statistiques concernant la longueur des champs, le nombre de sous-items et un résumé des définitions sont présentés sur l'écran du terminal. C'est le BOOKKEEPING.

3.2. Interrogation

Cette phase, commandée par le mot de contrôle INTERROGATIONS comporte aussi deux parties : les critères de sélection (RETRIEVAL CRITERIA) et les spécifications d'édition (EXTRACTIONS).

L'utilisateur détermine d'abord les expressions conditionnelles que devront satisfaire les éléments pour être sélectionnés, puis imprimés.

L'ensemble des critères est formé de conditions plus petites, indivisibles, reliées par des opérateurs logiques.

Une condition est généralement un sujet, représenté par le numéro d'item, suivi d'un prédicat.

3.2.1. Relations.

Le prédicat contient souvent deux parties : un opérateur de relation et une quantité de référence.

Les six opérations de base sont :

EQ	égal
NE	différent
GT	plus grand
LT	plus petit
GE	plus grand ou égal
LE	plus petit ou égal

Remarques :

- Lorsqu'un item est multiple, le sujet peut être le nombre de sous-items (TOTAL), la valeur la plus grande ou la plus petite, la somme ou la moyenne.
- La quantité de référence doit s'accorder en type avec le sujet.
- Avec les items alphanumériques, seuls les opérateurs EQ et NE sont autorisés. Leur signification est différente de la définition habituelle :

EQ	indique qu'une valeur, par exemple un titre, doit contenir une chaîne de caractères (valeur référence)
NE	indique le contraire

De plus un caractère spécial, le point, peut servir de masque et remplacer, lors de la comparaison, n'importe quel caractère non-blanc.

Exemple :

Nous cherchons les ouvrages dont le titre (item 6) contient le mot ORGANISATION. Mais les ouvrages écrits en anglais nous intéressent. Il faut donc prévoir l'orthographe différente ORGANIZATION.

La question est très simple, si on utilise le masque :

6 EQ *ORGANI.ATION*

Les astérisques autour de la valeur sont nécessaires comme délimiteur, puisqu'elle peut contenir des espaces.

3.2.2. Existence.

Des critères encore plus simples peuvent être utilisés : les critères d'existence. Ils sont d'ailleurs implicitement contenus dans la catégorie précédente. Ces conditions permettent de vérifier la présence ou l'absence de valeur pour un item déterminé.

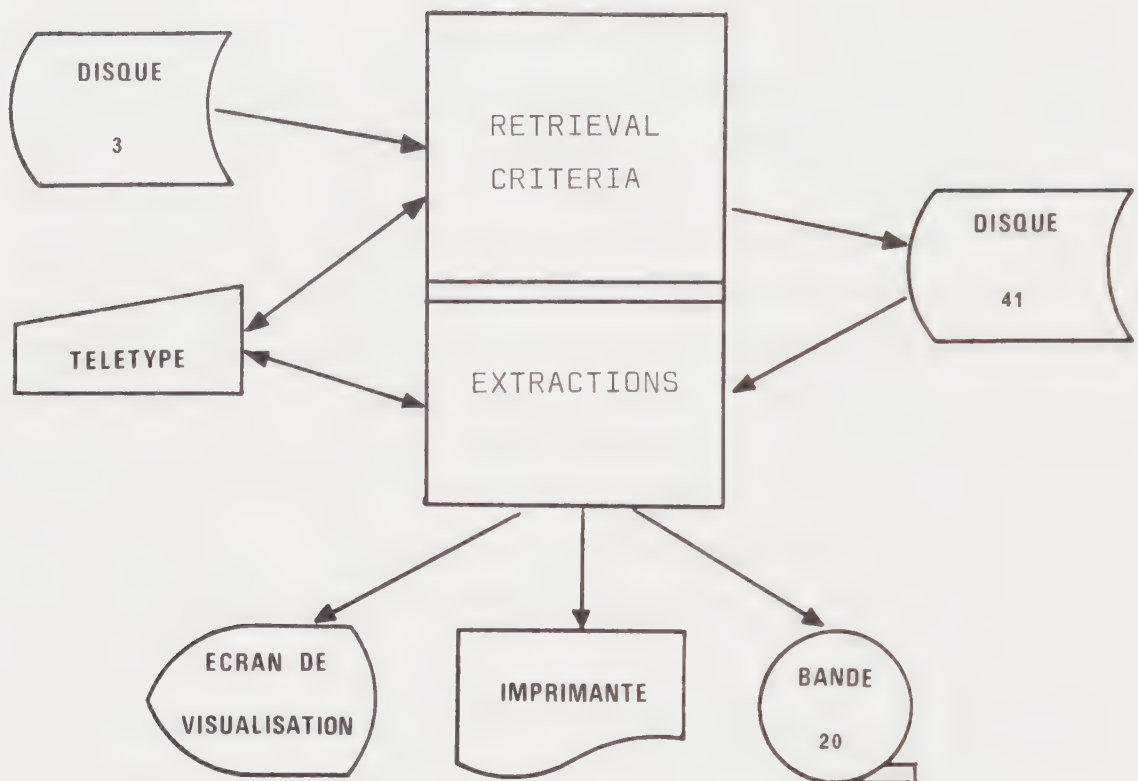
3.2.3. Expressions complexes.

Les critères peuvent être liés par des connecteurs logiques OR ou AND. Lors de l'évaluation, les conditions sont analysées de gauche à droite : les deux connecteurs ont la même priorité. Des parenthèses sont nécessaires si on désire changer l'ordre.

Exemple :

A AND B OR C	sera traité comme
(A AND B) OR C	ce qui est généralement
	différent de
A AND (B OR C)	

Aucune limite n'est imposée sur le nombre de parenthèses pourvu qu'elles soient appairées.



Les critères de recherche sont spécifiés à partir du clavier alphanumérique. La syntaxe est vérifiée. Le fichier est ensuite examiné séquentiellement suivant un algorithme défini dans le système. Le sous-ensemble des éléments sélectionnés est enregistré temporairement sur disque ou tambour.

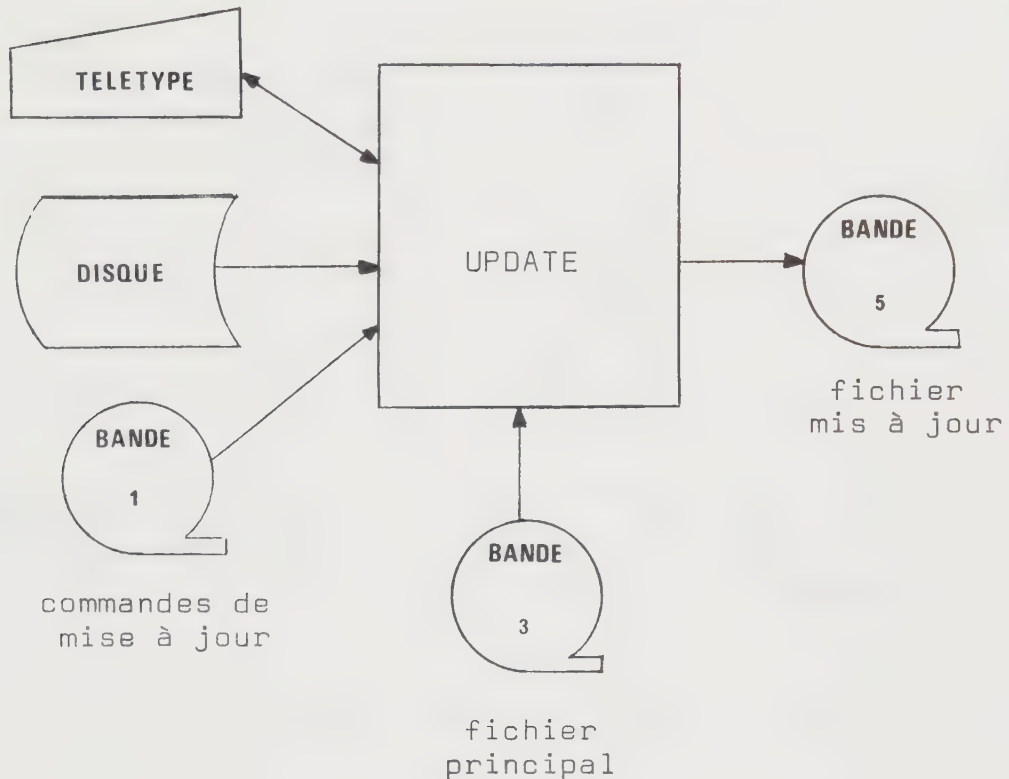
3.2.4. Edition.

Le sous-ensemble des éléments sélectionnés est ensuite trié, puis affiché sur l'écran du terminal et imprimé ou copié sur bande magnétique suivant la mise en page choisie.

Pour le même ensemble de critères de recherche, plusieurs rapports peuvent être générés dans des formats et sur des périphériques différents. Des options spéciales permettent d'écrire les données dans une forme adéquate pour être utilisées comme fichier d'entrée pour un autre ESTABLISHMENT, ou pour être traitées par un programme FORTRAN.

3.3. Mise à jour

Cette phase, commandée par le mot de contrôle UPDATE, est utilisée pour corriger certaines erreurs, ajouter, supprimer ou modifier l'information contenue dans le fichier.



Les commandes de mise à jour peuvent être préenregistrées sur bande magnétique ou disque. Les erreurs sont toujours affichées sur le terminal et doivent être corrigées immédiatement.

Seules les DISCRETE UPDATES sont implémentées, c'est-à-dire qu'il faut connaître le numéro de l'élément pour le modifier.

Alors qu'en mode SELECTIVE (existant seulement en version batch), des critères de recherche peuvent être appliqués au fichier de base pour sélectionner les éléments à modifier.

Dans une même session interactive, plusieurs passes de mise à jour peuvent être effectuées, seul le dernier fichier est conservé.

Les modifications sont possibles aux trois niveaux des données :

1	élément	NEW	:	ajouter
		REMOVE	:	supprimer
		MODIFY	:	modifier
2	item	INSERT		
		DELETE		
		SUBSTITUTE		
3	sous-item	ADD		
		ELIMINATE		
		REPLACE		

Pendant la mise à jour, toutes les validations imposées lors de la création du fichier sont appliquées aux nouvelles données.

Remarque :

Le traitement du fichier étant séquentiel, toutes les spécifications doivent être données selon l'ordre croissant des numéros d'éléments.

3.4. Autres fonctions

En plus des trois phases principales, une option permet d'obtenir la visualisation ou l'impression du BOOKKEEPING (voir paragraphe 3.1.). Elle est très utile lors du choix de la mise en page.

4. Format libre

Nous avons vu au paragraphe 2 une caractéristique essentielle de la structure de données utilisée par INFOL : l'enregistrement des données en longueur variable.

Un autre point fondamental est l'utilisation du format libre. La notion de carte perforée disparaît complètement. Il y a une seule exception lors de la création du fichier : l'utilisateur peut réserver un certain nombre de colonnes pour l'identification, par exemple la numérotation des cartes. Ce champ est ignoré par l'interpréteur, mais permet de réordonner un paquet de cartes déclassées ou de le corriger plus facilement : lorsqu'une erreur est détectée, le contenu de la carte est imprimé.

INFOL admet tous les caractères disponibles sur un ordinateur déterminé (BCD, EBCDIC, ASCII) sauf l'astérisque utilisé comme séparateur. Si le code utilisé est l'EBCDIC ou l'ASCII, la distinction est possible entre majuscules et minuscules lors d'une recherche en "texte libre".

Remarque :

Les mots de contrôle doivent être donnés en caractères majuscules.

Le blanc ou espace est aussi un délimiteur, comme dans le langage naturel, mais peut apparaître dans un texte alphanumérique.

Enfin, tous les mots-clés, déterminant ou précisant l'exécution d'une phase ne sont pas des mots réservés.

Exemple :

```
*6* EQ *UPDATE*
```

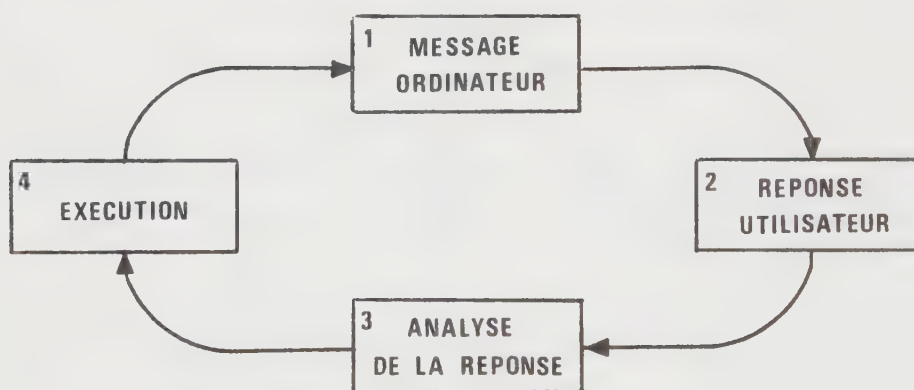
est une question correcte.

Chapitre IV : Programme interactif de gestion de fichiers

La manipulation de fichiers en traitement conventionnel (traitement par lots) présente plusieurs inconvénients, en particulier le délai entre la formulation des questions et l'impression des résultats.

Le dialogue direct avec l'ordinateur permet de réduire considérablement cette attente; de plus les erreurs sont détectées et peuvent être corrigées immédiatement, enfin le programme peut indiquer à chaque instant les options valables.

Mais la symbiose homme-machine, comme l'a définie J.C.R. Licklider [19], est particulièrement intéressante lors d'une interrogation du fichier : en fonction des résultats obtenus, l'utilisateur peut préciser ou élargir le domaine de sa recherche. Nous avons donc envisagé la transformation de l'interpréteur INFOL pour autoriser le mode conversationnel. Examinons maintenant les différentes étapes du dialogue homme-machine [4] :



Dès le début de la session interactive, le programme doit envoyer un message à l'utilisateur pour demander les renseignements nécessaires à la poursuite du travail, soit explicitement, soit implicitement. En effet l'interpréteur ne peut pas toujours poser une question précise à un instant donné; il indique alors par un caractère spécial que les informations précédentes sont correctes et enregistrées et qu'il attend la

suite de la commande : en INFOL, le format est libre, une instruction peut s'étendre sur plusieurs lignes.

Le programme attend ensuite une réponse. Mais le temps de réflexion varie énormément et sur un ordinateur monoprogrammé comme la CDC 3800, l'unité centrale est alors inactive. La version interactive d'INFOL mise au point sur cet ordinateur ne pourra devenir réellement opérationnelle qu'après l'introduction d'un système de ROLLIN/ROLLOUT. Un programme central peut théoriquement lire cette réponse de deux manières différentes : caractère par caractère ou ligne par ligne. Mais les routines d'entrée/sortie du FORTRAN ne travaillent que par l'intermédiaire du système d'exploitation, qui fait appel en fin de traitement à un petit programme, appelé "driver" servant d'interface software pour le périphérique concerné. Ce driver ne transmet la chaîne de caractères que lorsque la fin de message a été détectée : touche retour de chariot ou SEND, par exemple.

Le programme analyse ensuite la réponse du point de vue syntaxique et logique. Dans la forme la plus simple, on balaie la ligne caractère par caractère en sautant les espaces non significatifs. Si INFOL attend un mot clé, le mot tapé est comparé avec tous les mots acceptables à cet instant. S'il s'agit d'une donnée, elle est convertie en format interne, suivant son type.

Le programme apparaît plus intelligent et plus souple si les réponses anticipées sont admises. En effet, un nouvel utilisateur apprécie des questions détaillées mais dès qu'il acquiert de l'expérience, elles représentent une perte de temps assez considérable, surtout sur un télétype. Il est donc intéressant d'analyser toute la réponse de l'utilisateur et de poser une nouvelle question seulement lorsque le programme a atteint la fin de la ligne tapée ou en cas d'erreur.

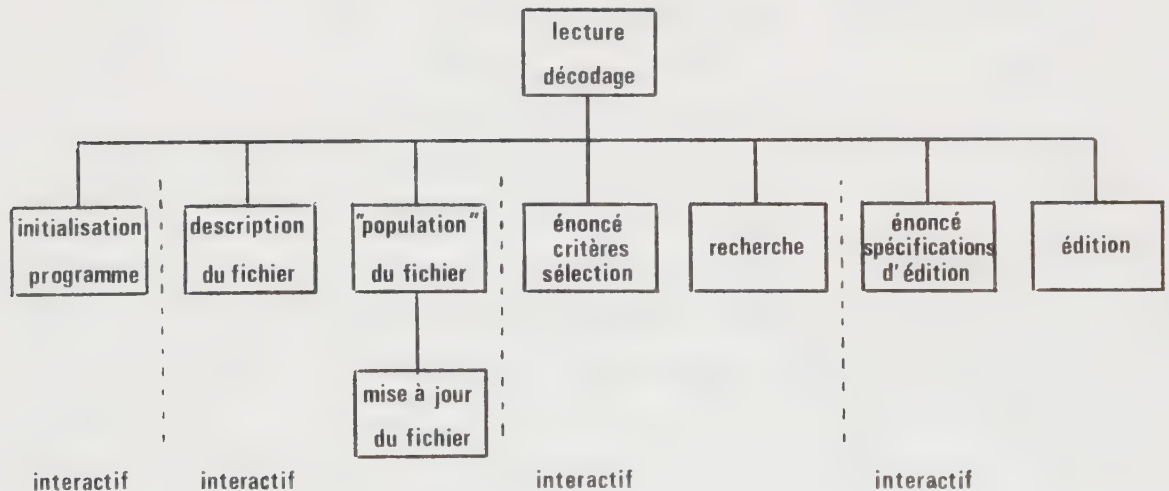
Si la réponse est correcte, on passe à la quatrième phase du cycle : elle peut se résumer à un stockage de sémaphores pour

une exécution future, mais elle peut aussi nécessiter la lecture de tout le fichier ou imprimer des centaines de lignes. Lorsque cette phase est terminée, le cycle recommence.

Les deux premières étapes représentent le dialogue homme-machine : le temps de réflexion est long et l'activité de l'unité centrale très faible. Par contre, pendant la quatrième étape, les possibilités d'intervention de l'utilisateur sont pratiquement inexistantes, mais le traitement de données peut être considérable.

1. Description générale de l'interpréteur

La division du programme en modules reflète parfaitement cette distinction entre les étapes fortement interactives et l'exécution proprement dite.



Il existe une seule exception : la population et la mise à jour du fichier : toutes les données sont testées, les erreurs détectées doivent être corrigées, mais le stockage des informations sur le fichier est exécuté dans le même module.

2. Routine d'initialisation DEBU

La version interactive d'INFOL a été écrite en FORTRAN standard USASI pour permettre un transfert aisé d'une machine à l'autre. Certains paramètres doivent cependant être définis différemment lors de l'installation sur un nouvel ordinateur. La routine DEBU attribue une valeur aux unités logiques et à certaines variables dépendant de la machine.

- unités logiques

DFILE	base de données
EFILE	fichier temporaire (résultat de la recherche)
QFILE=EFILE+1	} fichiers temporaires utilisés pour le tri
TFILE=EFILE+2	
VFILE=EFILE+3	
WFILE=EFILE+4	
SFILE	fichier temporaire utilisé pendant la création et la mise à jour du fichier
UFILE	fichier mis à jour
IIW	console en entrée
ILP	imprimante rapide
INP	fichier images-cartes pour création ou modification du fichier
IPUN	édition des résultats sur un medium différent de l'imprimante
ITV	console en sortie

- variables dépendant de l'installation

DATES	indicateur pour les dates	1	JJ/MM/AAAA
		2	MM/JJ/AAAA
IOW	terminal	-1	imprimante
		0	écran
		1	machine à écrire
KNB	décalage utilisé pour séparer l'adresse et la date d'entrée dans "R2"		
	CDC 6000	2 **	30
	CDC 3800	2 **	24
	IBM 360	2 **	20
	PDP 10	2 **	20
NC	nombre de caractères par mot-machine		
	CDC 6000	10	
	CDC 3800	8	
	IBM 360	4	
	PDP 10	5	
NIMAX	nombre maximal d'items par élément		
	éventuellement changement de vecteurs dans		
	COMMON / / W et K		
	COMMON /IO/ CD et FMT		
	COMMON /AN/		
	COMMON /INT/		
LIBUF	longueur des vecteurs "R2", "IBUF", "IBUF2"		
LELIST	longueur du vecteur "LIST" (longueur maximale en caractères pour les textes alphanumériques)		
LELYS	longueur du vecteur "LYS" (longueur maximale pour recherche en texte libre)		
LENGR1	longueur du vecteur "R1"		
PAGELS	longueur de page standard pour l'édition		
PAGEWS	largeur de page standard pour l'édition		
NBKAR	nombre de caractères dans la table de caractères		

INFOL envoie ensuite son premier message pour préciser les options par défaut :

```
IICE - GENEVA          INFOL-2 (VERS. 01/06/73)
OPTIONS = (D/M, PAGEWIDTH, LINES, TERMINAL)
          INFOL,D , 80      , 50      , DISPLAY
IF OK, TYPE THE PHASE YOU NEED, OTHERWISE WANTED OPTIONS
```

```
infol,d,80,60,type
```

La première information concerne les dates. Deux représentations externes sont possibles :

la notation américaine	MOIS/JOUR/ANNEE	→ option M
la notation européenne	JOUR/MOIS/ANNEE	→ option D

mais la convention choisie au début de l'exécution doit être respectée pour l'entrée de toutes les données. Par défaut, la convention européenne est utilisée (DATES=1).

Les deux indications suivantes donnent la taille de l'écran ou de la "page" du terminal; ce sont respectivement le nombre de caractères par ligne et le nombre de lignes par page. Nous avons ensuite établi une distinction entre les deux types de terminal : les messages envoyés seront plus courts sur les télétypes que sur les écrans de visualisation.

Trois options sont possibles :

```
DISPLAY
TYPEWRITER
PRINTER
```

La dernière n'a aucun sens sur un terminal, mais permet de soumettre un travail en "batch" avec la version interactive d'INFOL : il est possible de simuler le dialogue terminal-utilisateur sur cartes perforées et imprimante (test du programme).

L'utilisateur peut ensuite, s'il le désire, protéger son fichier par un "mot de passe" de deux mots-machine au maximum (8, 10, 16, 20 caractères suivant l'ordinateur) sans espace; sinon,

il indique directement la phase désirée

WRITE PASSWORD OR WANTED PHASE

pass bibliotheque

Pour lever les ambiguïtés possibles et faciliter la détection d'erreurs, le mot de passe doit être précédé du mot de contrôle PASSWORD.

Prenons l'exemple d'un fichier bibliothèque. Si la réponse est simplement BOOK, s'agit-il du mot secret de protection ou d'une demande de BOOKKEEPING ?

Un fichier protégé ne pourra jamais être relu si la comparaison révèle une discordance entre le mot enregistré et celui qui est donné. De plus, chaque tentative infructueuse est suivie immédiatement de la terminaison du job avec un message indicatif. Enfin, le mot est enregistré transcodé pour éviter la découverte par impression du contenu du fichier à l'aide d'un programme auxiliaire (DUMP).

Il est possible de spécifier un nouveau mot de passe lors de la mise à jour d'un fichier, mais non de le supprimer.

WRITE NEW PASSWORD OR WANTED PHASE

new pass bb2

Cette brève description permet de voir les caractéristiques essentielles que devront posséder les routines de lecture-décodage :

- distinguer les différents types de données : mot-clé, chiffre, ..., et détecter immédiatement les erreurs;
- les longues questions gênent un utilisateur expérimenté : les routines doivent être suffisamment souples pour reconnaître les réponses anticipées;
- l'utilisateur peut commettre certaines erreurs indétectables par INFOL, par exemple indiquer 6 comme longueur de page au lieu de 60. Une option spéciale doit lui permettre de revenir au début du module, nous l'avons appelée \$SOS.

3. Routine de lecture GET

Cette routine lit toutes les informations nécessaires au déroulement du programme, mais non au défilement des résultats sur l'écran (voir paragraphe 8).

Elle dispose d'un vecteur BUF(80) pour enregistrer une image-carte, à raison d'un caractère/mot. A chaque appel, le pointeur ICOL indique le prochain caractère à traiter dans cette zone. Lorsque celle-ci a été entièrement explorée : ICOL=81 et INFOL indique à l'utilisateur qu'il est prêt à recevoir de l'information en imprimant un signe =. Ce symbole a été choisi de préférence à : ou à ?, parce qu'il appartient à l'ensemble de caractères FORTRAN; il est donc disponible sur tous les ordinateurs, alors que le : par exemple est traité de manière spéciale sur CDC 6000. Suivant le jeu de caractères et suivant le périphérique utilisés, ce caractère est imprimé comme %. De plus, s'il apparaît dans un enregistrement BCD sur bande magnétique, il est interprété comme "End of Record".

Le pointeur ICOL, placé en COMMON, peut être modifié par d'autres routines dans deux cas :

- une question précise est posée par INFOL; il est inutile d'imprimer le signe =,
alors ICOL = 82;
- une erreur de logique ou de syntaxe a été détectée,
alors ICOL = 83.

Enfin, ICOL peut être égal à 84 si l'utilisateur a tapé l'option \$SOS.

Le paramètre de la sous-routine précise les caractères reconnus comme séparateurs lors de l'appel.

valeur du paramètre KEY	donnée attendue	séparateur reconnu	remarques
-1	nombre entier code	* ou espace	
0	mot-clé	* ou , ou espace	
+1	texte alphanumérique date nombre réel	*	
+2	chaîne pour recherche en texte libre	*	garde tous les blancs
+3	HEADLINE WITH TEXT	*	garde tous les blancs sauf le premier

Il est nécessaire de distinguer les trois derniers cas, bien que le seul séparateur reconnu soit l'astérisque.

- Pour la valeur d'un item, il ne faut conserver ni les blancs placés en tête ni ceux situés en queue ("leading and trailing").

KEY = 1

- Pour permettre la recherche de racines de mot, il est nécessaire de garder tous les blancs.

KEY = 2

Exemple : la question

6 EQ * GRAPHIC*

sélectionnera tous les éléments comportant le mot GRAPHIC dans le titre, mais non GEOGRAPHIC, CATALOGRAPHIC, ...

- Pour les options HEADLINE et WITH TEXT, le premier espace est considéré comme séparateur et n'est donc pas significatif.

KEY = 3

Deux compteurs permettent de connaître respectivement la longueur K de la chaîne de caractères analysée et le nombre de blancs terminaux à supprimer, car il est impossible de savoir si des espaces sont terminaux ou non avant d'avoir rencontré l'astérisque.

INFOL établit sur l'imprimante un journal des principales transactions effectuées sur le terminal, à l'exclusion des données entrées lors de l'INITIAL INPUT ou des NEW ELEMENT. Mais il est inutile d'imprimer les instructions erronées. Un sémaphore IFLAG indique si le contenu de la dernière "carte" lue doit être reporté ou non.

IFLAG = 0 supprime l'impression

IFLAG = 1 implique l'impression.

Une variable supplémentaire NX précise le nombre de colonnes réservées par l'utilisateur pour l'identification de ses cartes de données.

La possibilité de réponse anticipée a été introduite de la manière suivante : au terme de la reconnaissance par GET d'une donnée ou d'un mot-clé, le pointeur ICOL est-il égal à 80 ou reste-t-il encore des caractères non blancs sur la "carte" ? Dans l'affirmative, l'utilisateur a essayé de donner une réponse supplémentaire. Il suffira donc de tester la valeur de ICOL pour savoir si la question doit être posée ou s'il faut d'abord analyser cette réponse anticipée.

Le pointeur ICOL2 mémorise la valeur de ICOL, avant le commencement de ce balayage de la carte.

Nous avons adopté la méthode suivante pour le traitement des erreurs : un message précis est édité et, à l'exception de trois cas, l'utilisateur doit retaper toute l'information à partir de la donnée erronée. Ces trois cas sont les suivants :

- lors de la modification d'un élément, celui-ci dépasse la taille des zones réservées en mémoire. Il est alors néces-

saire de reprendre depuis le début de l'élément, sinon celui-ci est copié sur le nouveau fichier dans sa forme primitive;

- lorsque les items d'une association ne contiennent pas le même nombre de sous-items; l'utilisateur doit retaper toutes les données correspondant à cette association;
- enfin, lorsqu'un texte alphanumérique est trop long, tout le texte déjà tapé lors de l'impression du message est enregistré.

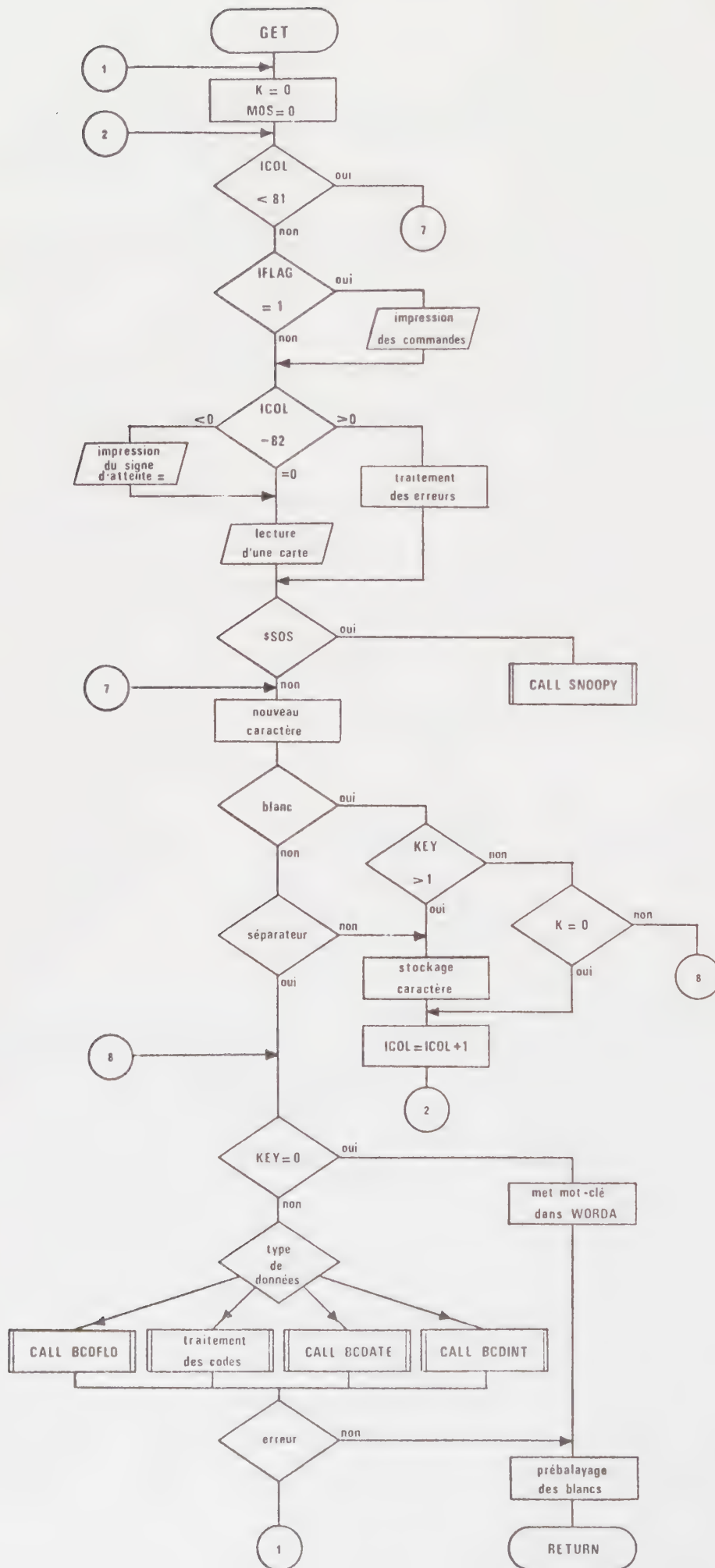
Pour diminuer le nombre de tests nécessaires sur la validité des données dans chaque phase d'INFOL, la routine GET effectue elle-même certains contrôles et réinitialise automatiquement une nouvelle lecture en cas d'erreur.

La variable KTYPE indique la nature de la donnée à analyser :

KTYPE = 1	nombre réel
2	texte alphanumérique
3	date
4	code numérique
5	code mnémonique
6	nombre entier
7	nombre entier

Dans les 6 premiers cas, la routine GET peut rencontrer un astérisque à la place d'une valeur. Au contraire avec KTYPE=7, il est nécessaire de trouver un nombre entier.

La figure 1. représente le diagramme fonctionnel de la sous-routine GET. Le listing correspondant est donné dans l'annexe I.



```
TICE - GENEVA          INFOL-2 (VERS. 01/06/73)
OPTIONS = (D/M, PAGEWIDTH, LINES, TERMINAL)
          INFOL,D , 80      , 50      , DISPLAY
IF OK, TYPE THE PHASE YOU NEED, OTHERWISE WANTED OPTIONS
infol,d,80,60,typewriter password essai1 establishment 4

NOW WRITE THE TEXT OF YOUR DESCRIPTION FOR EACH ITEM
FINISH EACH FIELD WITH AN ASTERISK
* 1*numero * prenom * date de naissance * profession *

TYPE THE CATEGORY TYPE AFTER ITEM NUMBER
FINISH YOUR FIELD WITH AN ASTERISK
* 1*unary numeric * multiple alphanumeric * unary date * unary coded *
```

Fig. 2. Exemple d'ESTABLISHMENT avec préréponse

```
TICE - GENEVA          INFOL-2 (VERS. 01/06/73)
OPTIONS = (D/M, PAGEWIDTH, LINES, TERMINAL)
          INFOL,D , 80      , 50      , DISPLAY
IF OK, TYPE THE PHASE YOU NEED, OTHERWISE WANTED OPTIONS
infol,d,80,60,typewriter password essai2 establishment 4

NOW WRITE THE TEXT OF YOUR DESCRIPTION FOR EACH ITEM
FINISH EACH FIELD WITH AN ASTERISK
* 1*numero * prenom *

* 3*date de naissance *

* 4*profession *

TYPE THE CATEGORY TYPE AFTER ITEM NUMBER
FINISH YOUR FIELD WITH AN ASTERISK
* 1*unary numeric *

* 2*multiple alphanumeric * unary date *

* 4*unary coded *
```

Fig. 3. Exemple d'ESTABLISHMENT avec préréponse partielle

```
TICE - GENEVA          INFOL-2 (VERS. 01/03/73)
OPTIONS = (D/M, PAGEWIDTH, LINES, TERMINAL)
          INFOL,D , 80      , 50      , DISPLAY
IF OK, TYPE THE PHASE YOU NEED, OTHERWISE WANTED OPTIONS
infol,d,80,60,typewriter

WRITE PASSWORD OR WANTED PHASE password essai3

WRITE NEW PASSWORD OR WANTED PHASE establishment

PLEASE, TYPE THE NUMBER OF ITEM DESCRIPTIONS YOU NEED 4

NOW WRITE THE TEXT OF YOUR DESCRIPTION FOR EACH ITEM
FINISH EACH FIELD WITH AN ASTERISK
* 1*numero*
```

4. Routine PHASE1

Cette routine traite la définition du fichier en quatre étapes :

ITEM DESCRIPTIONS

CATEGORY-TYPE

CODES

VALIDATIONS

Elle enregistre tous les renseignements dans une zone R1 de longueur LENGRI et 8 vecteurs de longueur NIMAX. Cet ensemble, appelé BOOKKEEPING (journal contenant les caractéristiques principales des données), est copié au début du fichier. La variable OVERL indique le numéro du module en cours d'exécution.

Période	Valeur d'OVERL
ITEM DESCRIPTIONS	21
CATEGORY-TYPE	22
CODES	23
VALIDATIONS	24
initialisation INITIAL INPUT	25

Ce découpage avec cinq valeurs différentes de OVERL est justifié par un gain de temps considérable lors de l'appel \$SOS.

Imaginons par exemple que l'utilisateur déclare un item UNARY plutôt que MULTIPLE. Aucune erreur n'est diagnostiquée, la syntaxe étant correcte; la seule méthode prévue pour revenir en arrière est \$SOS. Mais si une valeur unique de OVERL représente la PHASE1 toutes les descriptions devront être retapées.

Avec la solution choisie, une instruction GO TO calculé permet de revenir au début de la période en conservant toute l'information enregistrée dans les étapes précédentes. Enfin, à chaque retour de \$SOS, la présence du mot FINIS entraîne l'abandon de la phase ESTABLISHMENT (fig. 5).

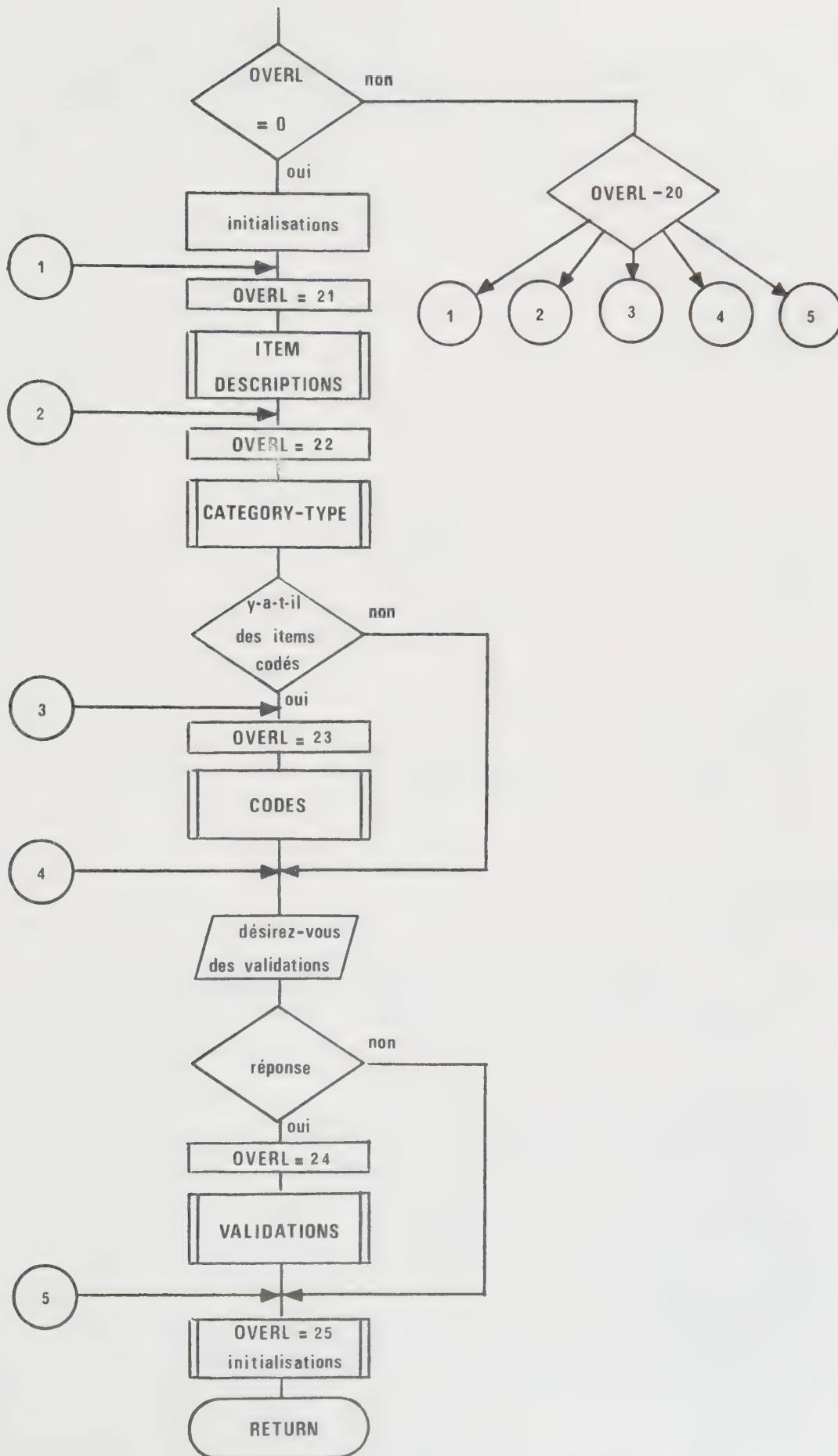
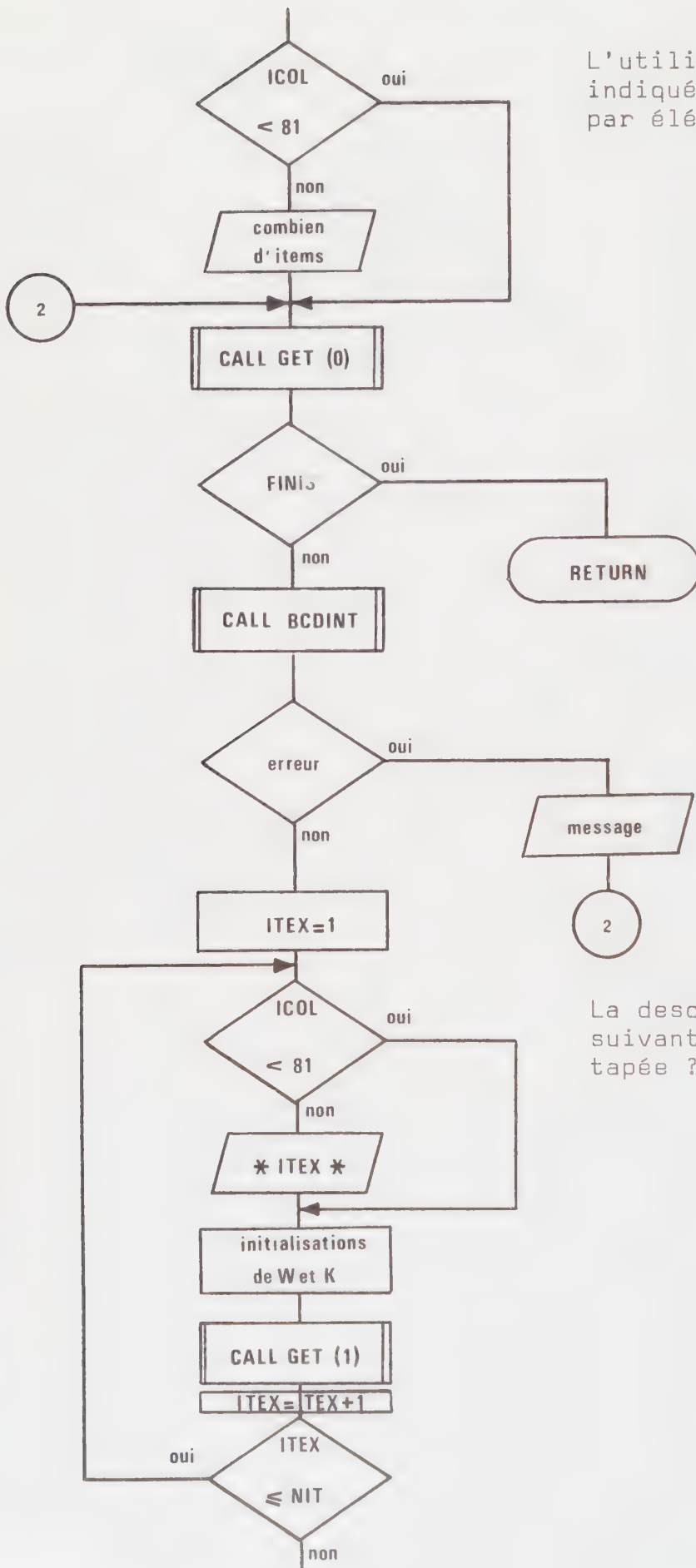


Fig. 5. Découpage de PHASE1 en modules



L'utilisateur a-t-il déjà
indiqué le nombre d'items
par élément ?

La description de l'item
suivant a-t-elle déjà été
tapée ?

5. Traitement des erreurs pendant l'INITIAL INPUT

La version interactive d'INFOL-2 admet comme source pendant cette sous-phase soit le terminal, soit un fichier image-cartes (unité logique 1).

Dans le premier cas, le traitement des erreurs est classique : dès qu'une erreur est détectée, elle doit être corrigée; tout le contenu de la ligne après l'erreur est ignoré et doit être retapé. A la fin de chaque élément, le message

ELEMENT nnn ACCEPTED

apparaît.

Dans le deuxième cas, en plus du message d'erreur, il est nécessaire d'imprimer la carte incriminée.

Deux solutions ont été testées :

- imprimer

ELEMENT nnn ACCEPTED

à chaque élément, ce qui permet de déterminer facilement le numéro de l'élément incorrect; mais si le fichier est important, 1000 lignes ou plus seront imprimées sur le terminal (machine à écrire).

- imprimer

LAST ELEMENT PROCESSED = nnn

seulement lorsqu'une erreur est détectée.

Dans tous les cas, la liste des numéros d'éléments acceptés est donnée sur l'imprimante rapide.

Nous montrons sur les figures 7 et 8 les deux méthodes. La bande magnétique a été créée à partir d'un ensemble de cartes. Sur certaines d'entre elles, le perforateur n'avait pas perforé la colonne 61. Certaines erreurs ne sont pas détectées : dans l'élément 51 (fig. 8), la lettre E de TRAITEMENT manque.

initial input 0 finis

ELEMENT 1 ACCEPTED
 ELEMENT 2 ACCEPTED
 ELEMENT 3 ACCEPTED
 ELEMENT 4 ACCEPTED
 ELEMENT 5 ACCEPTED
 ELEMENT 6 ACCEPTED
 ELEMENT 7 ACCEPTED
 ELEMENT 8 ACCEPTED
 ELEMENT 9 ACCEPTED
 ELEMENT 10 ACCEPTED
 ELEMENT 11 ACCEPTED
 ELEMENT 12 ACCEPTED
 ELEMENT 13 ACCEPTED
 ELEMENT 14 ACCEPTED
 ELEMENT 15 ACCEPTED
 ELEMENT 16 ACCEPTED
 ELEMENT 17 ACCEPTED
 ELEMENT 18 ACCEPTED
 ELEMENT 19 ACCEPTED
 ELEMENT 20 ACCEPTED
 ELEMENT 21 ACCEPTED
 ELEMENT 22 ACCEPTED
 ELEMENT 23 ACCEPTED
 ELEMENT 24 ACCEPTED
 ELEMENT 25 ACCEPTED
 ELEMENT 26 ACCEPTED
 ELEMENT 27 ACCEPTED
 ELEMENT 28 ACCEPTED
 ELEMENT 29 ACCEPTED
 ELEMENT 30 ACCEPTED
 ELEMENT 31 ACCEPTED
 ELEMENT 32 ACCEPTED
 ELEMENT 33 ACCEPTED
 ELEMENT 34 ACCEPTED
 ELEMENT 35 ACCEPTED
 ELEMENT 36 ACCEPTED
 ELEMENT 37 ACCEPTED
 ELEMENT 38 ACCEPTED
 ELEMENT 39 ACCEPTED
 ELEMENT 40 ACCEPTED
 ELEMENT 41 ACCEPTED
 ELEMENT 42 ACCEPTED

WORD EXPECTED IS * RATHER THAN 1968

US DE PROGRAMMATION ECONOMIQUE. *10*PARIS, DUNOD, 1968. *11 1968* *12*170 p. *1
 1968 *12*170 p. * 1

ELEMENT 43 ACCEPTED

ITEM NUMBERS MUST BE IN INCREASING ORDER

, COMPUTER, BUL. THE MYTHOLOGY OF EDUCATIONAL INNOVATION. *1 *CAMBRIDGE, HARVARD
 10*cambridge, harvard

ELEMENT 44 ACCEPTED

ELEMENT 45 ACCEPTED

ELEMENT 46 ACCEPTED

ELEMENT 47 ACCEPTED

NUMBER 12.000 IS OUT OF THE RANGE 1900.000 1980.000

ATA. *10*NEW YORK, LONDON (ETC.), J. WILEY, 1967. *11*1967* 12*585 p. *14** *1
 *12*585 p. *14** * 1

ELEMENT 48 ACCEPTED

Fig. 7. Exemple d'INITIAL INPUT à partir

d'une bande magnétique (méthode 1)

=initial input finis

WORD EXPECTED IS * RATHER THAN 1968

LAST ELEMENT PROCESSED = 42

UES DE PROGRAMMATION ECONOMIQUE. *10*PARIS, DUNOD, 1968. *11 1968* *12*170 p. *1
= *1968* *12*170 p. *1

ITEM NUMBERS MUST BE IN INCREASING ORDER

LAST ELEMENT PROCESSED = 43

, COMPUTER, RUN. THE MYTHOLOGY OF EDUCATIONAL INNOVATION. *1 *CAMBRIDGE, HARVARD
= 10*cambridge, harvard

NUMBER 12.000 IS OUT OF THE RANGE 1900.000 1980.000

LAST ELEMENT PROCESSED = 47

ATA. *10*NEW YORK, LONDON (ETC.), J. WILEY, 1967. *11*1967* 12*585 p. *14** *1
= *12*585 p. *14** *1

WORD EXPECTED IS * RATHER THAN BESH

LAST ELEMENT PROCESSED = 48

*01*49 *02*3.37 COM *03*COMPUTER METHODS IN THE ANALYSIS *04 BESHES, JAMES M. (
= *beshers, james m. (

ITEM NUMBERS MUST BE IN INCREASING ORDER

LAST ELEMENT PROCESSED = 48

D ED *10*CAMBRIDGE, LONDON, THE MIT PRESS, 1968. *11*1968* * 2*269 p. *14** *
= 12*269 p. *14** *1

AN ELEMENT MUST BEGIN WITH ITEM 1

1

LAST ELEMENT PROCESSED = 50

11*1967* *12*386 p. *13*INFORMATION SCIENCES SERIES. *14** *15*3.40**

SYSTEM EXPECTS AN INTEGER IN PLACE OF *

LAST ELEMENT PROCESSED = 50

*01*51 *02*3.42 DEW *03*DEWEZE, A. *04*DEWEZE, A. *06*TRAIT MENT DE L INFORMATI
= 01*51 *02*3.42 dew *03*deweze, a. *04*deweze, a. *06*traitement de l informati

ELEMENT NUMBERS MUST BE IN INCREASING ORDER

LAST ELEMENT PROCESSED = 50

01*51 *02*3.42 DEW *03*DEWEZE, A. *04*DEWEZE, A. *06*TRAITEMENT DE L INFORMATI
= 51 *02*3.42 dew *03*deweze, a. *04*deweze, a. *06*traitement de l imformati

ITEM NUMBERS MUST BE IN INCREASING ORDER

LAST ELEMENT PROCESSED = 50

12*214 p. *13*BIBLIOTHEQUE DE L AUTOMATICIEN, 15. *14** *1 *3.42*3.63*6.35**
= 15*3.42*3.63*6.35**

WORD EXPECTED IS * RATHER THAN PREN

LAST ELEMENT PROCESSED = 63

*01*64 *02*3.50 HAN *03*HANDBOOK OF SUCCESSFUL OPERATING *05 PRENTICE-HALL EDITO
= *prentice-hall edito

ITEM NUMBERS MUST BE IN INCREASING ORDER

LAST ELEMENT PROCESSED = 63

ORMS. *10*ENGLEWOOD CLIFFS, PRENTICE-HALL, 1969. *11*1969* * 2*767 p. *14** *
= 12*767 p. *14** *1

WORD EXPECTED IS * RATHER THAN 3.50

LAST ELEMENT PROCESSED = 64

THE RONALD PRESS CO., 1967. *11*1967* *12*665 p. *14** *15 3.50**
= *3.50**

Fig. 8. Exemple d'INITIAL INPUT à partir

deux bandes magnétiques (méthode 2)

6. Option \$SOS

L'utilisateur débutant a quelquefois des réactions imprévues lorsqu'un message d'erreur apparaît; il donne alors des réponses incorrectes, s'énerve et se perd rapidement dans un dédale de messages : il a finalement l'impression d'être dominé par la machine. Nous avons donc adopté un nouveau mot-clé \$SOS. Il permet à tout instant à l'utilisateur de repartir au début de la phase ou de la sous-phase en cours d'exécution.

Cette option est reconnue et traitée directement par la routine de lecture GET. Pour ne pas dégrader les performances et interdire cette combinaison de caractères, d'ailleurs peu probable, le test n'est effectué que sur les quatre premiers caractères de chaque ligne.

Le \$ a été choisi parmi les symboles spéciaux parce qu'il se trouve sur tous les terminaux, à l'exception peut-être de quelques terminaux français où le signe \$ est remplacé par ₣ (francs français).

La variable OVERL est associée à cette option : elle indique indirectement le module en cours d'exécution. Sa valeur est donnée par le numéro du segment primaire multiplié par 10 auquel on ajoute le numéro du segment de niveau 2 (voir chapitre V). Entre deux phases, sa valeur est nulle.

routine	phase	valeur d'OVERL
DEBU	initialisation	10
PHASE1	ESTABLISHMENT	21 à 25
P1P3	INITIAL INPUT	30
PHASE3	UPDATE	31
PHASE5	BOOKKEEPING	40
RETRIV	lecture des spécifications de recherche	50
RETASK	exécution de la recherche	60
XTRNPT	lecture des spécifications d'édition	70
PRESOR	tri	100
XTRXEC	édition	110
COMPLR	génération complète de rapport	111
COUIN	COUNT - INVERT	112
OTOMAT	génération automatique de rapport	113
WALK	"promenade" dans les résultats	120

La variable NPHASE indique la phase en cours d'exécution.

PHASE	valeur de NPHASE
initialisation	0
création	1
interrogation	2
mise à jour	3
bookkeeping	5

Entre deux phases, elle est égale à -1.

La première manière de réaliser cette option est de faire suivre chaque appel à GET d'un test pour examiner si \$SOS a été lu. Malheureusement il y a environ 120 appels différents dans l'interpréteur INFOL.

Nous avons choisi une autre solution : depuis GET nous rappelons le programme principal SNOOPY et celui-ci rappelle la phase désirée en fonction de la valeur de la variable OVERL.

Pour éviter d'appeler par un CALL un programme principal, ce qui n'est pas standard, nous avons scindé ce programme en deux parties :

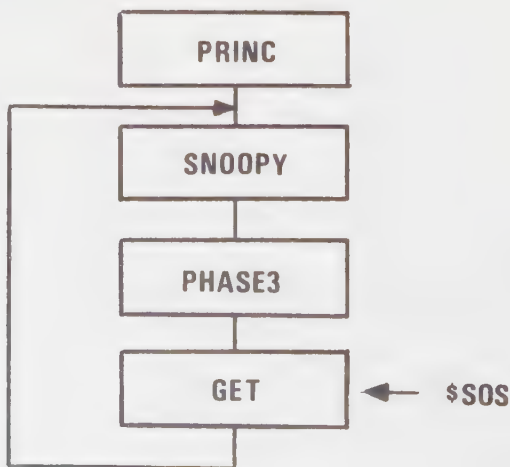
```
PROGRAM PRINC
COMMON /PHASE/ ... NPHASE ...
NPHASE = 0
CALL SNOOPY
END

SUBROUTINE SNOOPY
COMMON /PHASE/ ... NPHASE ...
IF (NPHASE.GT.0) GO TO 1
CALL DEBU
1 GO TO (2,3,4,6,5,6), NPHASE
...
CALL EXIT
END
```

Nous avons donc une structure pseudo-réursive.

Prenons l'exemple de l'utilisation de l'option \$SOS dans PHASE3.

On obtient le diagramme suivant :



Or en FORTRAN, lors d'un appel, un mot de la routine appelée contient l'adresse de retour. Mais il est unique, alors qu'en ALGOL une pile d'adresses est générée.

Lors du deuxième appel à SNOOPY par GET, la première adresse de retour est effacée et remplacée par la nouvelle. Il sera donc impossible de revenir dans PRINC, mais la fin normale du programme (CALL EXIT) étant placée dans SNOOPY, cela n'a aucune importance.

Seul le traçage en cas de fin anormale (erreur de parité, fin de fichier non testée, etc.) devient sans signification. Sur la figure suivante, nous présentons un tel traçage obtenu sur la CDC 3800.

UNCHECKED END OF FILE

*** ERROR OCCURRED ON TAPE NO - 42.

A =	0000000300113563	Q =	0000000300113563
ERROR DETECTED IN ROUTINE		I/O TSH.	
CALLED FROM	WALK	+	00200
CALLED FROM	PHASE2	+	00113
CALLED FROM	SNOOPY	+	00173
CALLED FROM	GET	+	00534
CALLED FROM	XTRNPT	+	00556
CALLED FROM	PHASE2	+	00066
CALLED FROM	SNOOPY	+	00173
CALLED FROM	GET	+	00534
CALLED FROM	XTRNPT	+	00556
CALLED FROM	PHASE2	+	00066
CALLED FROM	SNOOPY	+	00173
CALLED FROM	GET	+	00534
CALLED FROM	XTRNPT	+	00556
CALLED FROM	PHASE2	+	00066
CALLED FROM	SNOOPY	+	00173
CALLED FROM	GET	+	00534
CALLED FROM	XTRNPT	+	00556
CALLED FROM	PHASE2	+	00066
CALLED FROM	SNOOPY	+	00173
CALLED FROM	GET	+	00534
CALLED FROM	XTRNPT	+	00556
CALLED FROM	PHASE2	+	00066
CALLED FROM	SNOOPY	+	00173
CALLED FROM	GET	+	00534
CALLED FROM	XTRNPT	+	00556
CALLED FROM	PHASE2	+	00066
CALLED FROM	SNOOPY	+	00173
CALLED FROM	GET	+	00534
CALLED FROM	XTRNPT	+	00556
CALLED FROM	PHASE2	+	00066
CALLED FROM	SNOOPY	+	00173

7. Routine de sélection par clavier de fonctions

De nombreux mots-clés INFOL sont longs à écrire. Nous avons introduit la possibilité d'abrégier tous les mots par les quatre premiers caractères (longueur standard minimale sur toutes les machines à mots de plus de 32 bits) et même à 1 caractère pour certaines commandes de modification très fréquentes durant la mise à jour. Cette méthode présente deux inconvénients :

- sur le journal, ces instructions apparaissent telles quelles, c'est-à-dire en abrégé;
- la détection d'erreurs est moins facile.

Prenons un exemple : l'utilisateur désire remplacer la valeur de l'item 6. Il devrait écrire :

```
*6*  SUBSTITUTE ABC *
```

Mais à la suite d'une erreur de frappe ou de transmission, un espace supplémentaire est intercalé :

```
*6*  S UBSTITUTE ABC *
```

La lettre S est une abréviation correcte de la commande SUBSTITUTE, mais la valeur ajoutée sera UBSTITUTE ABC au lieu de ABC.

Pour gagner de la vitesse et éviter les inconvénients indiqués, nous avons décidé d'utiliser le clavier de fonctions disponible sur notre terminal. Cet équipement est divisé en deux parties :

- un système d'affichage lumineux permet d'indiquer à l'utilisateur les options disponibles à un instant donné, ce qui réduit encore les risques d'erreur;
- un clavier qui permet de demander une fonction.

Malheureusement, aucun blocage "hardware" n'empêche de choisir une fonction non sélectionnée (lumière éteinte). Tous les diagnostics devront être prévus dans l'interpréteur.

En entrée, une routine KBTAB [20], écrite en COMPASS, permet d'associer à chaque "touche" un mot d'un dictionnaire. Lors

de la phase d'initialisation d'INFOL, il est nécessaire d'effectuer l'appel suivant :

```
CALL KBTAB (DICKB,2)
```

où DICKB est le nom du vecteur-dictionnaire et 2 est la longueur des mots-clés (en mots machine).

En sortie, il existe deux possibilités pour allumer les lampes :

- appeler une routine en langage assembleur :

```
CALL KBLIGHT (0,1,2,3)
```

le nombre de paramètres est variable :

la valeur : 0 éteint d'abord toutes les lampes,
les valeurs ≠ 0 donnent ensuite le numéro des
lampes à allumer;

- remplir un masque de deux mots et l'écrire en FORMAT (2A8).

Pour améliorer la portabilité de l'interpréteur, nous avons décidé de grouper tous ces appels spéciaux dans une routine

```
SUBROUTINE KBSEL (IM)
```

Seuls, les appels à cette routine apparaissent dans l'interpréteur. Il suffit donc d'éliminer toutes les instructions de cette routine (sauf le RETURN) pour passer sur une autre installation. Le paramètre IM indique l'opération à effectuer.

IM = 0 appel à KBTAB

Pour allumer les ampoules, la première méthode nécessiterait un GO TO calculé à 24 branches, parce qu'il y a 24 situations différentes.

```
IF (IM.NE.0) GO TO 1
```

```
CALL KBTAB
```

```
RETURN
```

```
1 GO TO (2,3), IM
```

```
2 CALL KBLIGHT (0,1,2,3,4,5,9,10,12,13,15)
```

```
RETURN
```

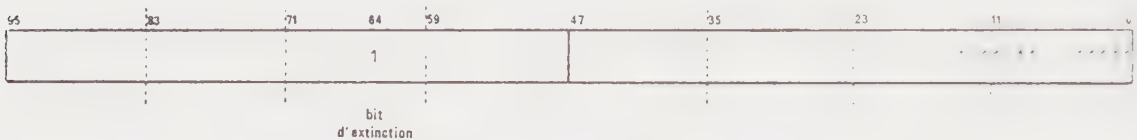
```
3 CALL KBLIGHT (0,4,5,9,10,12,13,15)
```

```
RETURN
```

La deuxième solution, plus rapide, a été choisie : IM et IM+1 donnent directement les adresses du masque et il suffit d'un seul ordre :

```
WRITE (8,2) KBMASK (IM), KBMASK (IM+1)
2 FORMAT (2A8)
```

Prenons l'exemple du masque nécessaire au début de l'exécution. Nous désirons allumer les ampoules : 1,2,3,4,5,9,10,12,13,15. Le numéro du bit à mettre à 1 est égal au numéro de la lampe -1.



La figure (9) donne une reproduction du masque transparent utilisé sur le clavier de fonctions : les mots clés sont disposés selon une structure arborescente : les commandes de phases en haut, les mots clés les plus fréquemment utilisés en bas. Les phases sont séparées par des traits épais, les groupes d'ampoules allumées simultanément dans une même phase par des traits moyens.

Exemple :

Pendant le message PLEASE WRITE THE PHASE YOU NEED les ampoules 10,12,13,15,16 sont allumées.

Si la phase UPDATE est choisie, toutes les lampes s'éteignent et seuls les numéros 20,21 s'éclairent : DISCRETE UPDATES et SELECTIVE CRITERIA.

La lampe 9 (\$SOS) est toujours allumée, puisque cette option peut être sélectionnée à n'importe quel moment.

Enfin, l'astérisque, bien que ne comportant qu'un seul caractère, a été placé sur le clavier de fonctions : cela permet de rester sur le même dispositif pour taper les CATEGORY-TYPE par exemple; de plus, sur le clavier alphanumérique, pour obtenir ce symbole, il est nécessaire de passer en majuscule.

1	INFOL	2	DISPLAY	3	TYPEWRITER	4	PASSWORD	5	NEW PASSWORD	6	YES	7	NO	8	*
9	SSOS	10	ESTABLISHMENT	11		12	UPDATE	13	BOOKKEEPING	14		15	INTERROGATIONS	16	FINIS
17	UNARY	18	MULTIPLE	19	NECESSARY	20	DISCRETE UPDATES	21	SELECTIVE CRITERIA	22	RETRIEVAL CRITERIA	23		24	EXTRACTIONS
25	ALPHANUMERIC	26	MNEMONIC	27	ALPHABETIC	28	REMOVE ELEMENT	29	SELECTIVE UPDATES	30	EXISTS	31	SUM	32	REPORT
33	CODED	34	NUMBER	35	NON-NUMERIC	36	NEW ELEMENT	37	DELETE	38	DOES NOT EXIST	39	MEAN	40	ELEMENT LENGTH
41	DATE	42	CODED AS	43	CHARACTERS	44	MODIFY ELEMENT	45	INSERT	46	ENTRY DATE	47	MINIMUM	48	SORTKEY
49	NUMERIC	50	INITIAL INPUT	51	INTEGER	52	ELIMINATE	53	SUBSTITUTE	54	ANY	55	MAXIMUM	56	COUNT
57	ASSOCIATION	58		59	MAXIMUM	60	ADD	61	REPLACE	62	SUB-ITEM ASSOCIATED	63	TOTAL	64	INVERT

8. Routine WALK

Sur les terminaux classiques (machines à écrire), les résultats sont imprimés, l'utilisateur peut lire et étudier en détail ses résultats. Par contre sur un display, lorsqu'on passe à la "page" suivante, la précédente est effacée. Nous avons voulu donner la possibilité de parcourir le fichier édité comme on feuillette un journal, c'est-à-dire sauter des pages, revenir en arrière, aller directement à la fin, etc.

La routine WALK est d'ailleurs utile avec un télétype où l'impression est lente : elle permet d'interrompre avec souplesse une sortie trop longue alors que la coupure définitive (BREAK) présente un grave inconvénient : non seulement l'impression en cours est stoppée, mais la question suivante posée par INFOL n'apparaît plus.

Cinq options sont disponibles :

- + permet de passer à la page suivante. S'il est suivi d'un entier n, on sautera n-1 pages. + est donc équivalent à +1. Un message d'erreur est donné si on dépasse la limite du fichier.
- réaffiche la page précédente. Un message apparaît si ce caractère de contrôle est utilisé alors qu'on examine la première page.
- R est une option rendue nécessaire par le système de mémorisation de l'image. Après un temps variable (5 à 20 minutes), la qualité de l'image se dégrade. R effectue le "rafraîchissement".
- B permet de revenir au début du fichier.
- E ignore la fin du fichier et redonne le contrôle pour une autre phase INFOL.

L'option DISPLAY, utilisée avec les spécifications d'édition, met le sémaphore LWALK à 1 dans la routine XTRNPT. Le traitement est ensuite normal, mais les résultats sont écrits sur

disque. La routine WALK est ensuite appelée. Elle simule d'abord l'entête d'une nouvelle page commençant par le caractère de contrôle 1H1. Ceci est rendu nécessaire par le fait que la dernière page contient en général moins de lignes que les précédentes et pour rester conforme au standard USASI, il est impossible de tester sur la "fin de fichier". Ainsi chaque page est contenue entre deux lignes commençant par 1H1.

Sur le disque, les enregistrements sont écrits en Format A, chacun précédé par un nombre indiquant sa longueur IFIN (en caractères). Ils seront relus par

```
READ (IP,1) IFIN, (PAGE(I), I=1, IFIN)
```

puis imprimés sur le télétype ou affichés sur l'écran

```
WRITE (ITV,2) (PAGE(I), I=1, IFIN)
```

En effet sur certains systèmes (INTERCOM) tous les blancs terminaux sont effectivement imprimés s'ils sont envoyés par l'ordre WRITE et nous avons voulu éviter cette perte de temps. La variable NPARD compte les pages lues, alors que la variable ICOUN indique le nombre de pages à sauter + 1.

Lorsque ICOUN est supérieur à 1, la page est à ignorer : elle est lue mais non écrite.

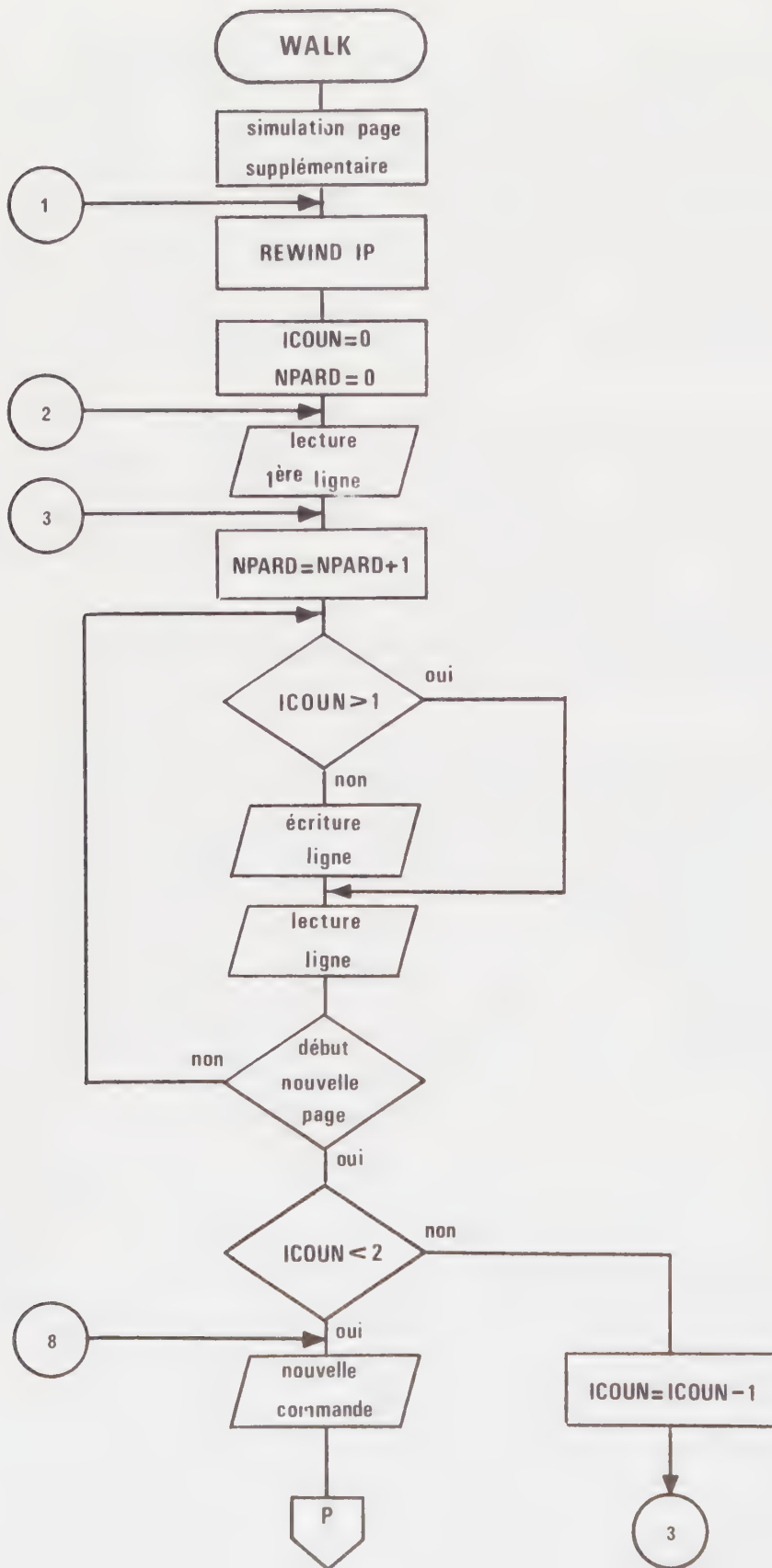


Fig. 10. Routine WALK

Traitement du fichier stocké sur disque

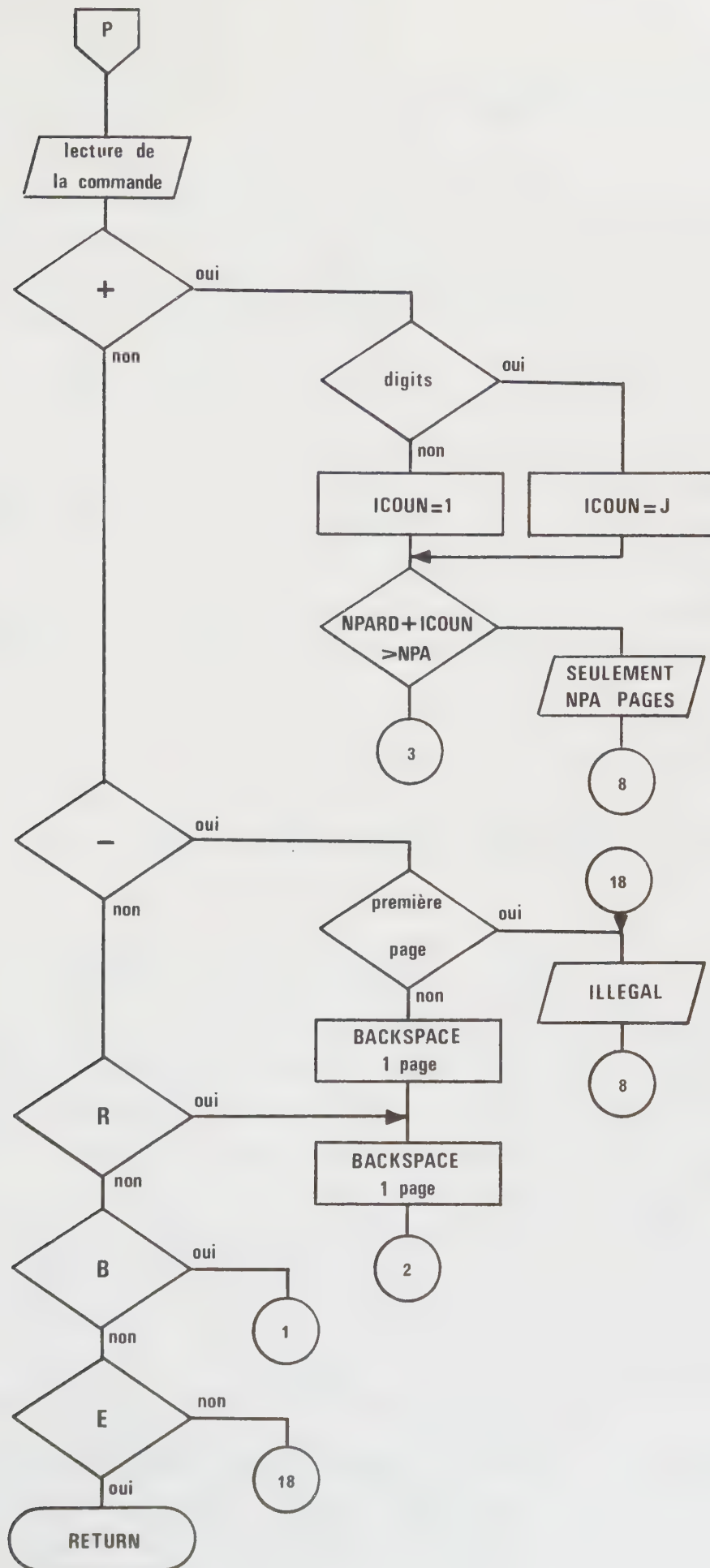


Fig. 11. Routine WALK : Traitement des commandes pour

9. Analyse statistique

Nous avons vu (chapitre III, paragraphe 3.2.3.) qu'INFOL permettait de spécifier, lors d'une interrogation, des critères complexes : chaque critère est composé d'un ou de plusieurs sous-critères reliés par des connecteurs logiques OR ou AND [18]. Mais dans le système original INFOL CDC, il n'est possible d'obtenir que le résultat global de la recherche.

Deux cas défavorables peuvent se présenter :

- aucun élément ne satisfait le critère global, mais comment déterminer le sous-critère ou l'ensemble de sous-critères qui a été trop sélectif ?
- beaucoup trop d'éléments non souhaités sont imprimés avec les autres.

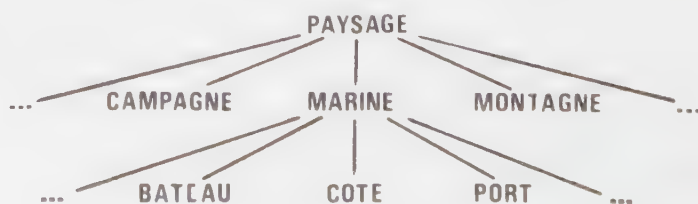
Nous avons introduit une analyse statistique qui donne une évaluation numérique des résultats des sous-critères :

- la fréquence absolue f_a , c'est-à-dire le nombre d'éléments qui ont satisfait un sous-critère déterminé;
- la fréquence relative

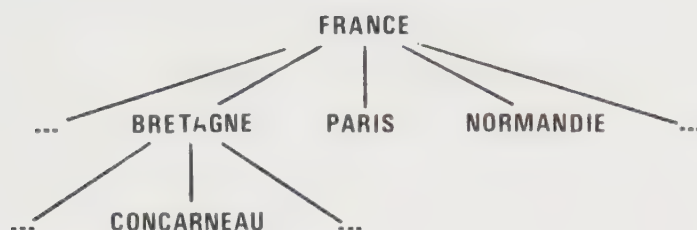
$$f_r = \frac{f_a \times 100}{\text{nombre d'éléments dans le fichier}}$$

A l'aide de cette analyse, l'utilisateur pourra d'une part éviter l'impression de résultats volumineux et inutiles et d'autre part cerner par des questions successives de plus en plus précises le résultat qu'il veut obtenir.

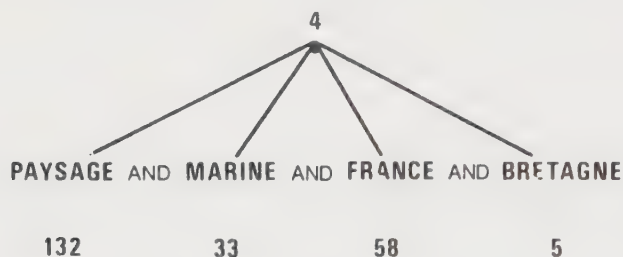
Interrogeons par exemple un fichier contenant des descriptions de tableaux. L'item 53 correspond au thème iconographique; cet item est construit à partir de descripteurs organisés de façon hiérarchique selon le modèle suivant :



L'item 52 décrivant le thème géographique est construit de manière semblable :



L'utilisateur recherche une demi-douzaine de tableaux; quatre questions successives lui permettent de sélectionner quatre tableaux (fig. 12 et 13). L'analyse statistique donne une évaluation numérique des différents ensembles correspondant à chacun des sous-critères et le résultat global de la question.



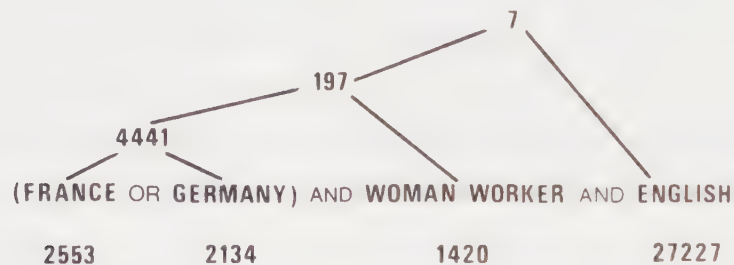
Lorsque l'utilisateur est satisfait, il peut imprimer la valeur de certains items; dans l'exemple, nous avons demandé l'édition du numéro d'enregistrement, le titre de l'oeuvre, le thème géographique et le thème iconographique.

Dans un système utilisant un fichier séquentiel, il faut consulter le fichier entier pour chaque interrogation. La création de sous-fichiers dans INFOL-2 permet de réduire considérablement le temps de recherche si certains critères sont très sélectifs : on crée un fichier ne contenant que les paysages (41%), puis à partir de celui-ci, un autre fichier avec seulement les paysages-marines, etc.

Malheureusement la méthode des sous-fichiers, si elle est efficace du point de vue temps, ne permet pas de revenir en arrière afin d'élargir à nouveau la question au cas où le

résultat global est nul.

Les fichiers séquentiels et les méthodes associées présentent ici une moins grande souplesse que les fichiers inversés. Avec ceux-ci l'analyse statistique permet de gagner du temps mais l'utilisateur a encore la possibilité d'élargir à nouveau les questions. Enfin, il est plus facile d'indiquer les niveaux intermédiaires de l'arbre fourni par les sous-ensembles ainsi créés (voir système ISIS du BIT) [21].



Dans l'exemple ci-dessus on recherche des documents en anglais traitant des femmes travaillant en France ou en Allemagne.

L'utilisation de ces différents types d'analyse statistique apporte un outil important dans les interrogations de fichier où le résultat final n'est que difficilement prévisible.

(Interrogation documentaire au sens large.)

ITCF - GENEVA INFOL-2 (VERS. 01/06/73)
OPTIONS = (D/M, PAGEWIDTH, LINES, TERMINAL)
 INFOL,D , 80 , 50 , DISPLAY
IF OK, TYPE THE PHASE YOU NEED, OTHERWISE WANTED OPTIONS
infol,d,80,35,typewriter

WRITE PASSWORD OR WANTED PHASE interrogations

ESTABLISHMENT 19/FEB/1973 LAST UPDATE 04/MAR/1973
-retrieval criteria 1

=*53* eq *paysage*

-extractions 1

STATISTICAL ANALYSIS

NUMBER OF ELEMENTS SATISFYING THE SEARCH = 132 (41.25 0/0 OF THE FILE)

=retrieval criteria 2

=*53* eq *paysage* and eq *marine*

-extractions 2

STATISTICAL ANALYSIS

NUMBER OF ELEMENTS SATISFYING THE SEARCH = 33 (10.31 0/0 OF THE FILE)

NUMBER	0/0 FILE	ITEM	SUB-CRITERION
132	41.25	* 53*	EQ PAYSAGE
33	10.31	* 53*	EQ MARINE

=retrieval criteria 3

=*53* eq *paysage* and eq *marine*

=*52* eq *france*

-extractions 3

STATISTICAL ANALYSIS

NUMBER OF ELEMENTS SATISFYING THE SEARCH = 58 (6.88 0/0 OF THE FILE)

NUMBER	0/0 FILE	ITEM	SUB-CRITERION
132	41.25	* 53*	EQ PAYSAGE
33	10.31	* 53*	EQ MARINE
58	18.13	* 52*	EQ FRANCE

Fig. 12. Exemple d'analyse statistique

```
=retrieval criteria 4
=*53* eq *paysage* and eq *marine*
=*52* eq *france* and eq *bretagne*
=extractions 4
```

STATISTICAL ANALYSIS

NUMBER OF ELEMENTS SATISFYING THE SEARCH = 4 (1.25 0/0 OF THE FILE)

NUMBER	0/0 FILE	ITEM	SUB-CRITERION
132	11.25	* 53*	EQ PAYSAGE
33	10.31	* 53*	EQ MARINE
58	18.13	* 52*	EQ FRANCE
5	1.56	* 52*	EQ BRETAGNE

```
=display *1* report *10* report *52* report *53* report finis
```

3/JUN/1973

PAGE

1

NUMERO	168
TITRE	PORT DU CROISIC. MATINEE D ETE
THEME GEOGRAPHIQUE	FRANCE. BRETAGNE. LE CROISIC
THEME ICONOGRAPHIQUE	PAYSAGE. MARINE. PORT

NUMERO	197
TITRE	QUAI A CONCARNEAU BATFAUX AMARRES
THEME GEOGRAPHIQUE	FRANCE. BRETAGNE. CONCARNEAU
THEME ICONOGRAPHIQUE	PAYSAGE. MARINE. QUAI

NUMERO	207
TITRE	CONCARNEAU HOTEL TRANSATLANTIQUE
THEME GEOGRAPHIQUE	FRANCE. BRETAGNE. CONCARNEAU
THEME ICONOGRAPHIQUE	PAYSAGE. MARINE. PORT

NUMERO	208
TITRE	CONCARNEAU VOILES AU SOLEIL
THEME GEOGRAPHIQUE	FRANCE. BRETAGNE. CONCARNEAU
THEME ICONOGRAPHIQUE	PAYSAGE. MARINE. BATEAU

DISPLAY COMMAND=e

END OF FILE

Fig. 13. Exemple d'analyse statistique et de
génération automatique de rapports

Chapitre V : Performances

Pendant ces trois années de recherche, nous avons trois objectifs qui paraissent au premier abord inconciliables :

- offrir à l'utilisateur des options plus nombreuses;
- diminuer la taille occupée en mémoire par l'interpréteur;
- accélérer l'exécution et principalement celle des phases peu interactives.

1. Définitions

Il est difficile de déterminer le temps de réponse moyen d'un programme interactif. En effet, pour toutes les périodes interactives, le temps de réaction dépend presque uniquement du système d'exploitation (monoprogrammation, multiprogrammation ou temps partagé) et, sauf dans le premier cas, du nombre d'utilisateurs à un instant donné, de la priorité, etc. Sur le système INTERCOM nous avons observé des temps de réponse compris entre 3 et 45 secondes. Par contre, pendant l'exécution des phases non-interactives, un tri par exemple, les performances dépendent beaucoup des algorithmes utilisés.

2. Règles générales d'optimisation

L'amélioration d'une routine ne doit pas se faire aux dépens des performances du reste du programme. Cela peut paraître évident, mais n'est pas toujours facile à réaliser. Nous avons donc cherché à écrire un programme modulaire, c'est-à-dire :

- formé de routines relativement petites, indépendantes et programmées de manière la plus linéaire possible. Elles sont ainsi plus faciles à tester, à modifier;
- paramétrisé au maximum, il est ainsi possible de modifier la longueur des zones de travail, la taille des enregistrements, la longueur du plus long message permis depuis le terminal, enfin et surtout la longueur des mots-machine.

Lorsque ces règles sont suivies, la portabilité du programme est facilitée.

Un autre objectif fixé est la minimisation du nombre d'accès aux fichiers, c'est-à-dire aux périphériques : en effet, dans un environnement multiprogrammé, le programme utilisateur perd le contrôle au profit d'un autre à chaque opération d'entrée-sortie.

3. Segmentation du programme

En général, la priorité d'un travail est inversement proportionnelle à la place mémoire utilisée. De plus, certains systèmes comme INTERCOM imposent une limite stricte aux programmes interactifs: ils ne doivent dépasser en aucun cas 60000₈ mots, c'est-à-dire 24576₁₀ mots de 60 bits, y compris les zones-tampons pour les opérations d'entrée-sortie.

Etudions la place exigée par INFOL :

zones de travail	3 x 2000	6000
tampons E/S	12 x 513	6156
vecteurs BOOKKEEPING	1000 + 11 x 150	2650
vecteurs auxiliaires	9 x 150	1350
		16156 ₁₀

Ce qui laisse 8420₁₀ mots pour le code du programme avec les routines d'entrée-sortie.

Nous avons dû segmenter l'interpréteur INFOL en modules ou OVERLAYS (fig. 14).

Le niveau 0 contient le programme principal, les routines de lecture et de décodage et toutes les variables communes à plusieurs overlays de niveau inférieur. L'option \$SOS a imposé d'y placer encore quelques variables particulières à un module, mais dont la valeur ne doit pas être effacée lors du rechargement du module.

Prenons un exemple : \$SOS est tapé pendant une mise à jour. Le programme PHASE3 sera rechargé au même endroit, mais toutes les variables temporaires seront détruites; il est donc indispensable de garder le numéro du dernier élément traité, les pointeurs de remplissage des zones de travail, etc.

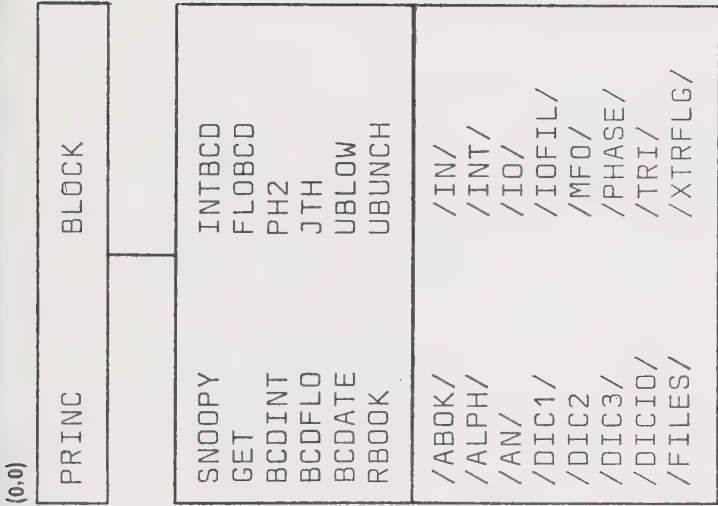
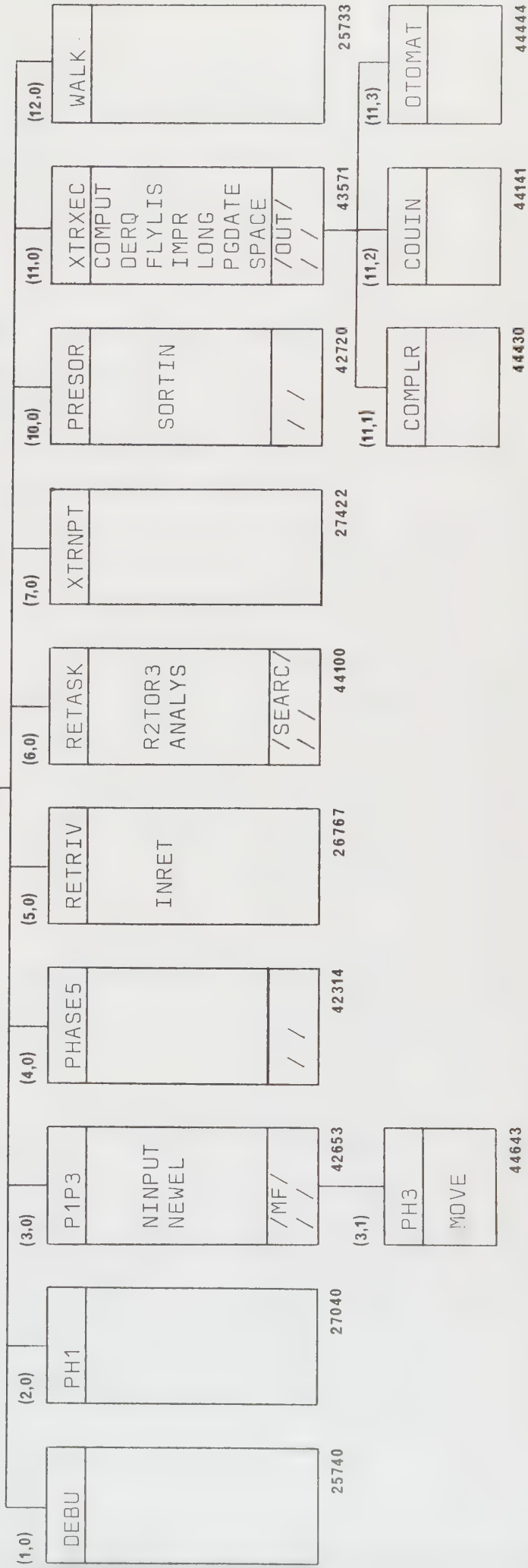


Fig. 14. SEGMENTATION

INFOL INTERACTIF

CDC 6400 (CERN)

SCOPE 3.3 FTN



Pour diminuer au maximum la place utilisée pendant les périodes interactives, les zones de travail sont placées en COMMON blanc. Comme elles sont justement inutilisées pendant l'exécution de la plupart des modules interactifs comme DEBU, PHASE1, RETRIV, XTRNPT, elles sont recouvertes par l'un des overlays de niveau 1.

Sur certains systèmes, il est possible de diminuer la taille des tampons d'entrée-sortie jusqu'à 63 mots, pour toutes les unités où le nombre d'opérations est faible et où la vitesse de transfert est peu importante : liaison avec le terminal, perforation de cartes, etc.

Avec cette astuce supplémentaire, l'espace nécessaire pour l'exécution est de 44634₈ positions (système d'exploitation SCOPE 3.3 sur CDC 6400).

Sous SCOPE 3.4, la situation est moins favorable, CDC ayant décidé d'incorporer aux routines standards d'entrée-sortie, les routines permettant de traiter les fichiers séquentiels-indexés et à accès aléatoire. L'espace nécessaire pour l'exécution devient alors proche de 53000₈ mots.

Toutes ces caractéristiques sont données pour un ordinateur CDC 6400.

4. Optimisation des opérations d'entrée-sortie

Nous avons particulièrement étudié 2 points :

- l'édition,
- le transfert des éléments.

L'interpréteur INFOL effectuait toute l'édition en format 136A1 : la portabilité du programme était excellente, puisque la longueur des mots n'intervenait pas. Mais l'analyse de la distribution du temps passé dans les différentes routines (SPY) nous a montré que la routine IOH. (traitement des enregistrements formatés sur la CDC 3800) était particulièrement coûteuse. Nous avons donc adopté une autre stratégie : un format variable est construit en fonction du nombre de caractères par mot et l'impression a lieu en FORMAT A4, A5, A8 ou A10

suivant l'installation. Nous observons alors sur la CDC 3800 un gain de 14 %.

Notre dernière étude a porté sur le blocage des éléments, selon la méthode indiquée au chapitre III.2.3.

A l'aide de la routine SPY (voir annexe II), nous avons comparé deux créations du fichier de notre bibliothèque (1500 éléments) : avec ou sans blocage.

Routine	Hauteur du canal		
	bloqué	non bloqué	
IOS.	299	886	vérifie les propriétés séquentielles des opérations E/S
IOB.	1017	513	traite les enregistrements avec FORMAT
IOP.	273	745	contrôle tous les appels E/S à SCOPE
IOH.	2999	3085	traite les enregistrements avec FORMAT
NEWEL NINPUT P1P3	532		routines INFOL de validation et de construction d'éléments
BCDFLO BCDINT GET	1734		routines INFOL de lecture-décodage
Temps total	198s	230s	

Nous observons le phénomène suivant : le temps passé dans IOS. et IOP. est plus important dans le cas non bloqué, ce qui est normal car il y a plus d'appels. Par contre, l'inverse se produit (dans des proportions moindres) avec IOB. : en effet, l'attente pour la fin d'opération a lieu dans cette routine et lorsque les blocs sont plus longs, l'attente est plus grande.

5. Traitement des interrogations complexes

Nous avons remarqué que le temps de recherche augmentait très rapidement avec la complexité de la question posée, c'est-à-dire avec le nombre de sous-critères.

Une simulation a été effectuée sur le fichier bibliothèque comportant 1500 éléments à l'aide d'une question sans réelle signification, mais présentant une particularité : aucun élément ne pouvait y répondre. Nous supprimons ainsi les opérations annexes (copie d'élément, etc.), ce qui simplifiait l'analyse de l'histogramme fourni par SPY (fig. 15).

La question posée peut se résumer ainsi :

$$\begin{aligned} & C1 * (C2 * C3) * C4 * ((C5 * C6) + (C7 * C8)) \\ & * ((C9 + (C10 * C11)) + (C12 * C13)) * C14 * C15 \end{aligned}$$

en représentant l'opérateur AND par * et OR par +.

Nous constatons la situation suivante : le traitement des connecteurs logiques (canaux 1-12, 72, 73, 90-93) représente 52 % du temps (unité centrale).

La méthode utilisée était identique à celle de Northwestern pour leur première version d'INFOL-2 : chaque connecteur logique était traité l'un après l'autre suivant sa priorité. A chaque étape l'expression était reformée avec un critère de moins.

Exemple :

$$\begin{aligned} & C1 * (C2 + C3) \\ \rightarrow & C1 * C2' \\ \rightarrow & C1' \end{aligned}$$

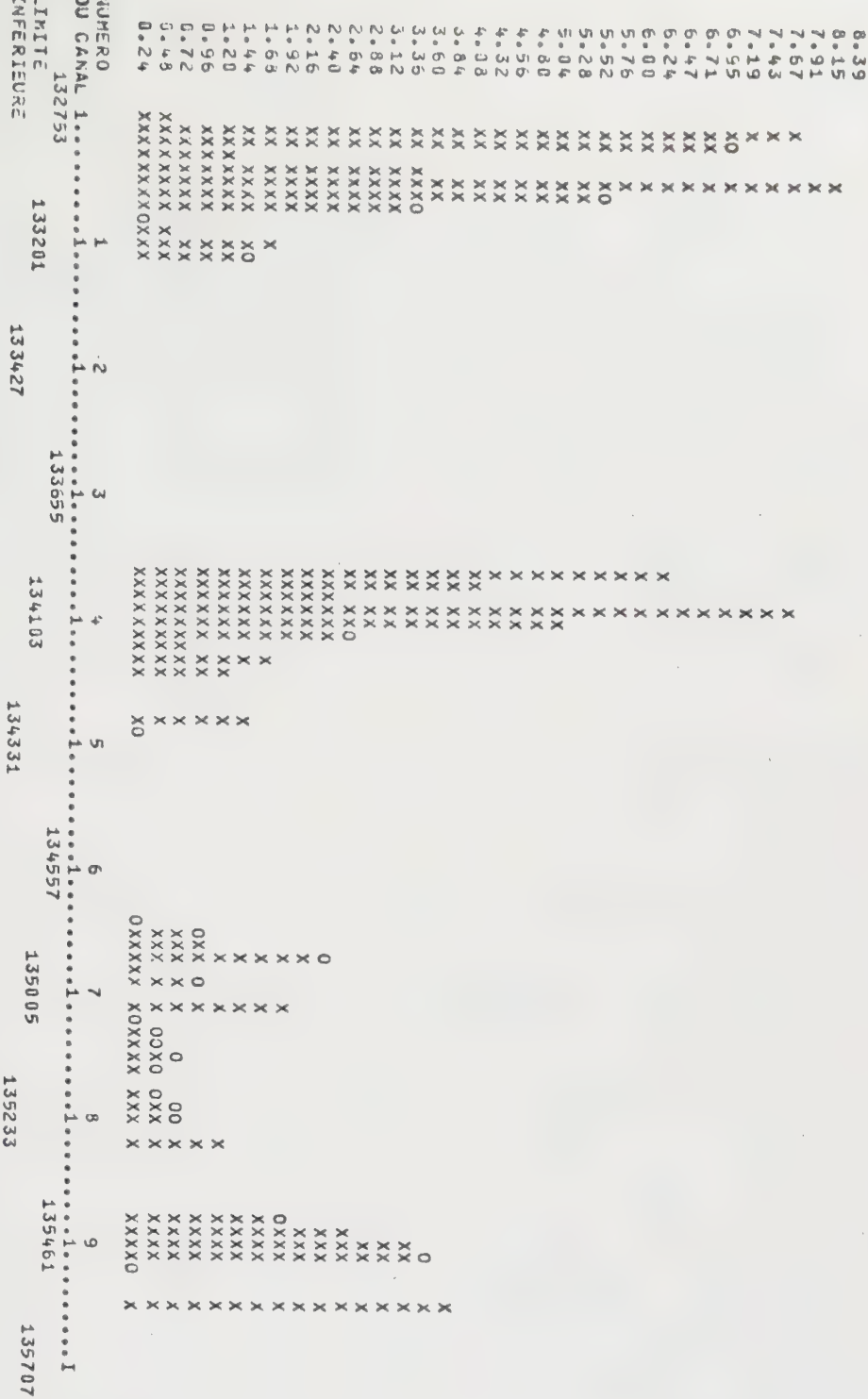
La méthode présente 2 inconvénients :

- les recopies de conditions sont lentes,
- l'expression est toujours analysée en entier, alors que dans l'exemple précédent, si C1 est faux il n'est pas nécessaire de continuer.

Nous avons donc essayé d'adapter la méthode présentée par WOLPE [22]. L'idée est la suivante : on construit avant de

Fig. 15. HISTOGRAMME DES TEMPS (méthode 1)

POURCENTAGE



TRUTH

RETASK

canal
opération

lecture DFILE

initialisation

critère d'existence

recherche valeur externe

type

calcul nombre de sous-items

initialisation sous-critère associé

recherche valeur externe enregistré résultat

sous-critère

reste-il encore des sous-critères associés ?

comptage des sous-items satisfaisant les sous-critères associés

transfert dans vecteurs W9 et W10 de OMEGA et W11

appel à TRUTH

un nombre suffisant de sous-items a-t-il satisfait la recherche ?

critères de relation sur réels

critères de relation sur entiers

éclatement des α

comparaison des champs α

incrémentatation compteurs

recopiage des vecteurs OMEGA, W11, P

appel à TRUTH

appels à OMSGI (traitement de la liste de variables du READ)

37
38
39
40
42
43
45
49
66
67
68
69
70
72
73
74
75
76-79
80-83
84
85-87
89
90-92
93
96

6. Les programmes interactifs et le FORTRAN

Nous avons utilisé le FORTRAN pour l'écriture de notre compilateur pour deux raisons essentielles :

- une première version d'INFOL-2 avait été programmée avec ce langage supérieur,
- la portabilité des programmes est bonne.

Mais nous avons rencontré plusieurs difficultés lorsque nous sommes passés à la version interactive.

Le premier problème est l'interface entre le FORTRAN et le terminal : il n'existe pas sur toutes les installations. En particulier, nous n'avons pu effectuer pour cette raison de test sur la CII 10070 sous le système SIRIS 7.

L'exécution est difficile à interrompre par la frappe d'un caractère sur le terminal : il est impossible d'utiliser l'instruction READ; lorsqu'une opération READ est initialisée, tout le programme est interrompu jusqu'à l'arrivée effective du caractère. Par contre, l'instruction BUFFER IN (non standard) répond parfaitement aux objectifs, malheureusement il est impossible de lire tantôt avec READ, tantôt avec BUFFER IN sur la même unité.

La réalisation de l'option \$SOS est rendue délicate : en effet, le Fortran USASI ne génère pas de pile pour stocker les adresses de retour des routines, ce qui interdit les appels pseudo-récurifs.

Enfin, le FORTRAN est mal adapté au traitement de caractères et ne permet pas l'allocation dynamique.

Mais comparons le à d'autres langages [23] :

le compilateur COBOL ne peut générer de code réentrant et de plus, la compilation et l'exécution sont lentes. Le PL-1 par contre peut produire des programmes réentrants, malheureusement la portabilité est réduite. Enfin le langage assembleur est difficile à tester et à maintenir.

Chapitre VI : Conclusions et perspectives

L'expérience que nous avons poursuivie nous amène aux conclusions suivantes :

- 1) La transformation du programme non interactif en un programme conversationnel a demandé un effort de trois hommes-année au maximum, temps passé en grande partie à améliorer les performances.
- 2) La portabilité d'un interpréteur interactif tel qu'INFOL est excellente, car nous avons évité d'utiliser des instructions ou des gadgets trop spécifiques à une installation.
- 3) L'utilisation de fichiers séquentiels rend la programmation simple et les performances restent acceptables pour les fichiers de petite ou moyenne importance (quelques milliers d'éléments). Pour des applications de plus grande envergure, le temps d'interrogation devient prohibitif; les fichiers inversés deviennent pratiquement indispensables, mais la mise-à-jour doit pratiquement être effectuée en "traitements par lots".
- 4) Le succès d'INFOL auprès des utilisateurs est essentiellement dû à la simplicité du langage de commande. Cette condition devrait être respectée, même si l'organisation des fichiers était changée (accès aléatoire ou séquentiel-indexé).

Annexe I : Listing de la sous-routine GET

```

SUBROUTINE GET (KEY)
C-----
C      SUBROUTINE READS INPUT STRING (COMMAND OR NEW DATA)
C-----
      COMMON /ALPH/ LELIST,LELYS,LIST(780),LYS(145)
      COMMON /DICIO/ COMMA,DICN(14),DIGIT(10),FAC(10),PLUS,POINT,SEP(5)
      COMMON /IN/ DATES,DAY,EFLAG,ICOL,IFLAG,K,KF1,KP2,KTYPE,N,NX,WORDA
      COMMON /IO/ BLANK,CD(300),FINIS,FMT(300),IDAT(11),KAL,MOINS,NC,ND,
1PAGE(136),PAGELS,PAGEWS,STAR
      COMMON /IOFIL/ IIN,IIW,ILP,INP,IOW,IP,IPUN,ITV
      COMMON /MFO/ INEL,INTEG,IST,ITO,ITO2,PREV
      DIMENSION BUF(80)
      INTEGER B,BLANK,BUF,COMMA,DAY,DOLLAR,PREV,SECOUR,STAR,WORDA
      LOGICAL EFLAG
      DATA DOLLAR/1H$/ ,SECOUR/4H$SOS/ ,ICOL2/82/
C-----
C      KEY = -1  READ UNTIL STAR OR BLANK
C      KEY = 0   READ UNTIL STAR, BLANK OR COMMA
C      KEY = +1  READ UNTIL STAR
C      KEY = +2  READ UNTIL STAR KEEPING THE BLANKS
C      KEY = +3  READ UNTIL STAR KEEPING THE BLANKS, EXCEPT THE FIRST
C
C      ICOL<81  NORMAL PROCESSING
C      ICOL=81  NEXT CARD
C      ICOL=82  QUESTION ASKED BY INFOL. NO EQUAL SIGN PRINTED
C      ICOL=83  ERROR IN INPUT
C      ICOL=84  AFTER $SOS
      IF (KEY.GT.1) ICOL=ICOL2+KEY-2
1    MOS=0
      K=0
2    IF (ICOL.LT.81) GO TO 7
      IF (IFLAG.EQ.1) WRITE (ILP,29) BUF
      IF (K.GT.LELIST) GO TO 27
      IF (ICOL-82) 4,5,3
C      ERROR IN INPUT
3    IF (IIN.EQ.INP.AND.ICOL.NE.84) WRITE (ITV,30) PREV,BUF
      IF (IOW.GE.0) WRITE (ITV,34)
      READ (IIW,32) BUF
      EFLAG=.FALSE.
      ICOL=1
      GO TO 6
4    IF (IIN.EQ.IIW.AND.IOW.GE.0) WRITE (ITV,34)
5    READ (IIN,32) BUF
      ICOL=NX+1
6    IF (BUF(1).EQ.DOLLAR) GO TO 26
7    B=BUF(ICOL)
      IF (B.NE.BLANK) GO TO 9
      IF (KEY.GT.1) GO TO 11
      IF (K.EQ.0) GO TO 12
      IF (KEY) 17,14,8
8    MOS=MOS+1
      GO TO 11
9    IF (KEY.NE.0) GO TO 10
      IF (B.EQ.COMMA.OR.B.EQ.STAR) GO TO 13
      GO TO 11
10   IF (B.EQ.STAR) GO TO 15
      MOS=0
11   K=K+1
      LIST(K)=B
12   ICOL=ICOL+1
      GO TO 2
13   IF (K.EQ.0) GO TO 16

```

```
14  L1=K
    IF (K.GT.4) L1=4
    CALL UBUNCH (LIST,WORDA,L1)
    GO TO 24
C    KEY = 0    PACK INTO -WORDA- FOR DICTIONNARY COMPARISON
C    KEY ≠ 0    RETURN WITH NUMERIC AND DATE VALUES IN -WORDA-
C                OR ALPHANUMERIC TEXT IN -LIST-
15  IF (K.NE.0) GO TO 17
    IF (KTYPE.EQ.7) GO TO 20
16  WORDA=B
    ICOL=ICOL+1
    GO TO 24
17  K=K-M0S
C    STORE DIFFERENT TYPES OF DATA
    GO TO (22,18,21,20,19,20,20), KTYPE
18  WORDA=BLANK
    GO TO 24
19  IF (K.GT.NC) GO TO 28
    CALL UBUNCH (LIST,WORDA,K)
    GO TO 24
20  CALL BCDINT
    WORDA=N
    GO TO 23
21  CALL BCDATE
    WORDA=DAY
    GO TO 23
22  CALL BCDFLO (KX)
    WORDA=KX
23  IF (.NOT.EFLAG) GO TO 24
    ICOL=83
    GO TO 1
C    POST-SCANNING OF BLANKS (ANTICIPATED RESPONSE)
24  ICOL2=ICOL
25  IF (ICOL.EQ.81) RETURN
    IF (BUF(ICOL).NE.BLANK) RETURN
    ICOL=ICOL+1
    GO TO 25
C    PROCESSING OF $SOS
26  CALL UBUNCH (BUF,WORDA,4)
    IF (WORDA.NE.SECOUR) GO TO 7
    ICOL=84
    CALL SNOOPY
C
27  WRITE (ITV,31) LELIST
    GO TO 17
C
28  WRITE (ITV,33) NC,(LIST(M),M=1,K)
    ICOL=83
    IFLAG=0
    GO TO 1
C
C
C
29  FORMAT (1X,80A1)
30  FORMAT (25H LAST ELEMENT PROCESSED =,I8/2X,80A1)
31  FORMAT (13H TEXT EXCEEDS,I4,50H SYMBOLS BUT YOUR ALREADY TYPED TEX
1T IS ACCEPTED )
32  FORMAT (80A1)
33  FORMAT (23H MNEMONIC CODE EXCEEDS ,I2,12H CHARACTERS ,30A1)
34  FORMAT (2H =)
    END
```

Annexe II : Description de la routine SPY

SPY permet l'impression des histogrammes du temps passé dans les différentes régions d'un programme.

SPY se compose d'un sous-programme FORTRAN pour l'initialisation et l'impression des histogrammes et d'une routine COMPASS (Assembleur 3800) qui effectue l'échantillonnage du temps et remplit les différents canaux de l'histogramme.

Trois paramètres doivent être définis lors d'un appel d'initialisation :

CALL SPYIN (LDEB,LSTEP,IPER)

LDEB est l'adresse inférieure de la région à espionner.

LSTEP est le nombre d'adresses couvertes par un canal.

La zone couverte par SPY est de $100 \cdot LSTEP$.

IPER est la période d'échantillonnage en millisecondes.

Le principe de la routine SPY est le suivant : le programme est interrompu à intervalles réguliers; on examine alors le compteur d'instructions et on ajoute une unité au contenu du canal correspondant.

Un appel à HISTSPY interrompt l'échantillonnage, le contenu des canaux est normalisé. L'histogramme est ainsi toujours imprimé sur une seule page. Il se présente de la manière suivante : selon l'axe vertical, on donne le pourcentage du temps passé dans les différentes régions examinées par SPY. Selon l'axe horizontal, les adresses (exprimées en octal) permettent à l'utilisateur de repérer, en consultant la table de chargement de son programme, les routines ou les groupes d'instructions correspondant aux différents canaux.

Annexe III : Bibliographie

- [1] Vallée, J.; Chalice, R.; Mittman, B.
Implementation of a retrieval language in FORTRAN. A progress report on INFOL-2.
Evanston, Ill., Northwestern Univ., 1968.

- [2] Chenais, J.M.
Etude de la portabilité de INFOL-2. Langage de création, d'interrogation, d'édition et de mise-à-jour de fichiers (Thèse de doctorat).
Genève, CICE, 1973.

- [3] Baud, J-Ph.; Jacquesson, A.
Un système interactif pour la création et l'interrogation de fichiers.
In : Revue suisse d'économie et de statistique.
Berne, SSSEP; 107 (1971) 3, pp. 627-628.

- [4] Meadow, C.T.
Man-machine communications.
New York, J. Wiley, 1970.

- [5] Davis, R.M.
A history of automated displays.
In : Datamation.
Los Angeles, BPA; 11 (1965) 1, pp. 24-28.

- [6] Van Dam, A.
Computer driven displays and their use in man-machine interaction.
In : Advances in computers.
New York, Academic press; 7 (1966), pp. 239-290.

- [7] Machover, C.
Graphic CRT terminals characteristics of commercially available equipment.
In : AFIPS conf. proc. Fall Joint computer conf.
Montvale N.J., AFIPS press; 31 (1967), pp. 149-159.

- [8] Tubes et dispositifs à images.
Paris, Thomson-CSF, 1972; (GTE 021).

- [9] Wear, L.L.
An interactive Keyboard for man-computer communications.
In : AFIPS conf. proc. Spring Joint computer conf.
Montvale, N.J., AFIPS press; 36 (1970), pp. 607-612.

- [10] Levrat, B.; Vuilleumier, G.; Wenger, M.
Terminal graphique conversationnel.
In : Nouvelles techniques.
Zürich, ASSPA; (1972) 4, pp. 111-114.

- [11] Proposed USA standard COBOL.
In : SIGPLAN notices.
New York, ACM; 4 (1967) 2.

- [12] Dodd, G.G.
Elements of data management systems.
In : ACM Computing Surveys.
New York, ACM; 1 (1969) 2, pp. 117-133.

- [13] Feature analysis of generalized data base management systems.
CODASYL systems committee technical report.
New York, ACM, 1967.

- [14] Chapin, N.
Common file organization techniques compared.
In : AFIPS conf. proc. Fall Joint computer conf.
Montvale N.J., AFIPS press; 35 (1969), pp. 413-422.

- [15] Roberts, D.C.
File organization techniques.
In : Advances in computers.
New York, Academic press; 12 (1972), pp. 115-174.

- [16] Knuth, D.E.
The art of computer programming. Vol. 1: Fundamental algorithms.
Reading, Addison-Wesley, 1968.

- [17] INFOL Reference Manual 3600/3800.
St. Paul, CDC, 1966; (Publ. N° 60170300).

- [18] INFOL-2 Manuel de référence. 2ème éd.
Genève, CICE, 1973.

- [19] Licklider, J.C.R.
Man-computer Symbiosis.
In : IRE Transactions on human factors in electronics.
Washington, IRE; HFE-1 (1960) 1, pp. 4-11.

- [20] Piguet, R.
Implantation d'un terminal sur un ordinateur de la deuxième
génération (Thèse de doctorat).
Genève, CICE, 1973.

- [21] Schieber, W.D.
Technical manual on ISIS. A generalized information storage
and retrieval system designed at the ILO.
Stockholm, Statskontoret, 1972.

- [22] Wolpe, H.
Algorithm for analyzing logical statements to produce a
truth function table.
In : Communications of the ACM.
New York, ACM; 1 (1958) 3, pp. 4-13.

[23] Yourdon, E.

Design of on-line computer systems.

Englewood Cliffs, Prentice-Hall, 1972.

[24] Harms, J.; Jacquesson, A.

SPY - Histogramme du temps utilisé dans un programme.

(L300 - Sept. 1972).

In : Bibliothèque de programmes. Genève, CCI, 1968.

